HOW TO TEACH A NEW ROBOT NEW TRICKS - AN INTERACTIVE LEARNING FRAMEWORK APPLIED TO SERVICE ROBOTICS.

A Dissertation
Presented to
The Academic Faculty

by

Sekou L. Remy

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
October 2009

HOW TO TEACH A NEW ROBOT NEW TRICKS - AN INTERACTIVE LEARNING FRAMEWORK APPLIED TO SERVICE ROBOTICS.

Approved by:

Dr. Ayanna M. Howard, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Charles Kemp
Health Systems Institute
Georgia Institute of Technology

Date Approved: December 2009

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The applications of robotics are changing. Just as computers evolved from the realm of research and extreme novelty tools to now becoming essential components of modern life, robotics is also making a similar transition. With the changes in applications come changes in the user base of robotics. These users will span a broad range of society, but there are some key properties that can be used to characterize them. First, they more often than not will not be the designers of the robots. Second, they will not have robot control as their primary task while operating the robot. Third, they will not have the resources or the desire to provide all the training that the robot will require, yet they will have the need to fine tune robot performance to their specific needs. Fourth, they will want to use multiple modes of interaction to make the robot accomplish the primary task. Fifth, they will expect and demand that the robot remain safe at all times (safe to humans, pets, or personal property) and expect the robot to be a readily replaceable appliance (cheap). Sixth, they will expect that the robot will be intelligent, at least in the confines of the task at hand.

These are some of the key properties that will exist for the new user base. To address some of the needs that will arise because of these properties, we propose work that enables behavior transfer from teacher to robotic student that is facilitated through observation and interaction. Many users in the projected user base will not have exposure to the technologies that enable robotic operation. These users will however have some degree of understanding of how they would like the robot to provide assistance in accomplishing the task. The goal of this work is specifically to enable the user to transfer this understanding to the robot, and have the robot acquire this understanding via interactive learning.

To make interactive learning possible via interaction we believe that the robot will have to be able to perform some degree of self regulation. Further, since it is assumed that the user will not have access to the robot's internal mechanisms, the robot will also have to be able to properly manage the knowledge it acquires over time and to verify and validate its understanding periodically. Scaffolding, a method in which teachers provide support while the student learns to master portions of a task, is likely to be the primary method to facilitate this process.

This research will undertake study of coherence and its relevance to learning by observation. It will also implement the components that would enable a robot to learn to perform a small set of tasks and demonstrate them in various settings. For this work a robot will be defined as a hardware platform upon which a software agent operates. It is our desire that this software agent will be equipped to operate on any platform and learn any task that a human could perform with the same resources.

Demonstrations of this research will highlight service robotics, with emphasis on assistive applications.

CHAPTER I

INTRODUCTION

The 21^{st} century has brought many changes to human society. Many of the images of the future projected from the 20^{st} century have become common place, while many unexpected changes have also become realities. One area that has captivated the minds of many, robotics, still lies on the threshold, not yet to the point where automobiles, microwave ovens and airplanes have progressed.

Robotics, a word derived from the Czech word *robota* - meaning forced labor, was initially popularized by its use in a 1920's play R.U.R. written by Czech author and playwright Karel Capel [16]. Without philosophical analysis of the play, or its author, it can be presented that a robot is indeed a device that is designed for human need.

The International Federation of Robotics (IFR), a global body created to stimulate and manage the field of robotics, categorizes robots into two classes: Industrial and Service. Industrial robots are those designed to facilitate manufacturing operations while service robots perform services useful to humans. The class of service robots can further be subdivided into professional service robots and personal/private service robots. Professional service robots are those used by companies to provide services to their customers while personal/private service robots are used by individuals themselves.

In a recent study [40] the IFR assessed development and growth of the robotics industry. The study found that there were 31.6k units of professional service robots and 2.9M units of personal/private service robots presently deployed. These numbers, cumulative up to 2005, indicate how many service robots have already been purchased and are currently in use. An interesting projection also included in the study is that

in the 2006-2009 time frame it is expected that some 34k units of professional service robots (an increase of almost 200%) and 5.6M personal/private service robots (an increase of over 100%) will be sold.

Such projected increases, especially in the personal/private service robotics class, pose an interesting challenge for the robotics industry. As the number of robots increases, it is almost inevitable that the number of robot users will also increase. Increases in the numbers of users, especially users of subclasses like personal/private service robots, would radically change the overall demographics of robot users.

Even as the workplace grows more technical with each generation [67, 88], it is clear that typical users of service robots will not have the specific technical expertise and experience as that of robot designers. For the projected sales of these robots to be viable, these users will have to be endowed with mechanisms to control and possibly adapt the behaviors of their robots. Such capability would permit factory programmed robots to be optimized for their deployed environment in a manner that satisfies their owner/operators. It can also be the case that the technical users would benefit from an approach that permits them to perform control and modification tasks without the need for familiarity with the details of the robotic platform.

For these disparate types of users, the *challenge* remains the same: devise a method that permits control of a robot by users who will likely not have the time, expertise or exposure to write complex code or to modify robotic control circuitry.

In the following sections, a brief introduction to relevant work will be presented that should equip the reader with key concepts and applications that have been designed or implemented in the past to address this *challenge*.

1.1 On the Robot

In the previous section, the origin of the term robot was presented and classes of robots were identified. In this section, the actual definition of what a robot actually is will

be provided along with a light treatment of core robotic concepts in this definition.

1.1.1 Definition

For this research we use as a foundation Bekey's definition of an autonomous robot. In his work [6] he defines a robot as a machine that senses, thinks, and acts and is capable of operating in the real-world environment without any form of external control for extended periods of time. While a computer may be a building block of a robot, Murphy rises to note that a robot differs from a computer in that it can interact in the physical world [61]. The definition of the phrase *extended periods of time* would surely make for a lively debate by many researchers, yet Bekey acknowledges that there are indeed varying degrees of autonomy. It is also important to highlight that Bekey mentions that these systems perform some function or task that may be intended by their human creator or may be an unexpected, emergent behavior.

The assertion that the human creator is a prime generator of a robot's function has long been maintained, and this is one of the assertions that we believe must be challenged in this new age of robotics. Specifically we believe that the robot ought to be able to become a more active participant in its learning process. We recognize that the robot's motivation for its action should still be directly related to human need or human command, but greater separation between this motivation and need will be needed for robots to become more useful.

To capture the varying modes of humans, Scholtz [86] defined the following roles for users interacting with robots: supervisor, mechanic, and peer. And in a later work the role of the bystander [87] was added to the group of roles. In our estimation, the role of mechanic as defined in [86] is essentially an operator role where the operator has knowledge comparable to the creator of the robot. We believe that these two roles, operator and designer, are distinct roles and in this new age of robotics, must be treated as such. It will no longer be acceptable to assume that a robot operator

has knowledge about the robot that is equivalent to its designer. In many cases it is this robot operator who will be expected to provide significant input as far as the robot's capabilities and there should be a mechanism for such input to be provided. It should also be noted that when considering service robotics, the operator may or may not be the owner or even the creator of the robot.

1.1.2 Fundamental Mechanisms

Sensing, thinking and acting lie at the core of the definition of a robot. As such, understanding each of these, and how they relate is invaluable in understanding of the robot. Sensing is the process by which the robot detects properties of the environment, or of itself, through the use of devices called sensors. Such devices are designed to detect specific physical properties and are restricted in range of operation, sensitivity, and precision. Sensors provide the only mechanism for a robot to "detect" anything about its environment.

Acting is the process of changing the environment in some way through the use of devices called actuators that are connected to the robot. Actuators cause controllable change in the environment or in the robot. These devices provide the only mechanism for the robot to actively generate changes in the environment or in its own configuration. Common examples of actuators include motors and pistons. Actuators are the devices that enable differentiation between a robot and a computer since computers are considered to be devices that generate output data, and do not create physical change in the world.

Finally, thinking refers to the process that is performed on sensing data with the aim of determining the proper action, or sequences of actions that are required to attain the current goal(s). While sensing and action are closely restricted by the physical hardware available to the robot, clever processing (thinking) can be performed on sensor data to improve its quality, and usefulness [72].

1.1.3 Purpose

For robots to be useful in many of the target applications and to provide the needed assistance, a robot is expected to display human-like intelligence. To this end, in the recent past, robots have been designed to play chess [100], play air hockey [7], navigate through the Mojave desert [96] and even though an urban cityscape [94]. Robotic fish [49] and receptionists [32] have also been deployed. In essence, the general descriptions of common robotic function are either the 3D's (dangerous, dirty, dull), innovative, or cool. Examples of the latter group include the amazingly lifelike android Repliee Q2 [60], robotic artists [41] and Qrio, the actor/singer/dancer/golfer [29].

1.2 On the Process of Robotic Control

Now, with the term *robot* more fully defined and described, and with a brief outline of some robotic applications we now move to the question of what avenues exist to implement such applications.

There have been several fronts through which applications of robotics have been implemented. Spanning academic disciplines, philosophies, motivations and applications, a large number of solutions have been generated. In this section some of these solutions or approaches are presented. While often referred to as stand alone approaches, we caution the reader that in reality, many researchers combine these (and other approaches). In addition, while wide acceptance of certain approaches has varied over time, there in no linear progression in time indicated by the order in which these approaches are presented.

1.2.1 Teleoperation

Teleoperation, and the linked technology *telerobotics*, are approaches that support physical action over some distance [33]. Teleoperation permits the operator to directly guide and cause each incremental motion of the robot, while telerobotics utilizes

higher level communication abstraction and instead of causing incremental motion, the operation sets goals and approves (or modifies) plans generated by the robot.

In both cases, the human ultimately controls the operation of the robot and uses the robot's sensory information for decision making. Examples of these systems are documented in [27] which features a mode that permits gesture based teleoperation of a Pioneer mobile robot, the work presented in [80] which demonstrates a method for diagnostic ultrasound, and [62] which discusses lessons learned during search and rescue.

1.2.2 Scripting

Monitoring the robot's progress in the manner required by teleoperation is not always feasible or realistic. Especially in situations where high latency or delay exists in either sensing or action, a different approach is often required; one such approach is scripting.

Scripting involves encoding an algorithm that the robot executes with code generated using some programming language like LISP, C++, Java or Assembly language. Knowledge is captured through the use of programming constructs such as sequence, repetition, branching and procedures. Using these constructs, an understanding of the problem (robotic task), and access to sensor and actuator data, code is written to control the robot's operation. Expert system solutions like [17] and [97] which are implemented in this manner perform well in static or well characterized environments.

Fuzzy [50] and probabilistic [44] approaches have also been applied to scripting to release programmers from some of the restrictions of explicit parameters and complex relationships between data values.

The fact that the robot is able to operate on its own once the script is executing enables it to be an autonomous agent linked to the human only when code or data

is being transferred to or from it. Neither incremental motion, nor goals are explicitly influenced by the operator once the code is running, thus this approach differs significantly from teleoperation/telerobotics.

Before concluding the discussion on scripting, mention will be made of Mathematical Programming. While not programming in quite the same sense as scripting, mathematical programming [103], or in one of its more popular forms *Linear Programming*, has been applied to robotics [59], manipulation [98], as well as to computer vision [93]. This type of programming is essentially a process of optimizing (either minimizing or maximizing) a real function. The canonical form of Linear Programming is presented in Equation 1

$$\begin{aligned} &\text{min} && z = c^T x \\ &\text{s.t.} && Ax \leq b \\ &\text{where} && x \geq 0 \end{aligned} \tag{1}$$

Where z is the objective function to be optimized, $Ax \leq b$ defines the feasible space of solutions, and x is the vector of variables that define the solution. While the form of this type of programming differs greatly from scripting, developing solutions in this manner shares many of the same processes. Dynamic Programming and Integer Programming are other key types of Mathematical Programming that have been applied to this domain.

1.2.3 Finite State Automata and Hybrid Automata

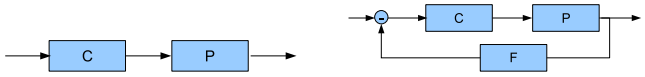
Finite State Automata (FSA) and Hybrid Automata (HA) are two classes of systems that can be used to define how a robot executes actions. Whether Mealy type or Moore type, these automata describe the execution as a finite number of states and transitions between these states. In many implementations, each state is linked to an action (actuation) and sensors are linked to the transition between actions. HAs

differ from FSAs in that they provide affordance for the inclusion of temporal data in the system.

Like scripting, these approaches require explicit knowledge of the environment and the target application at design time for the robot to execute the proper action. Unlike scripting however, proofs and theorems can be leveraged to provide guarantees or even bounds on performance [4, 23].

1.2.4 Classical Control

Also leveraging a strong mathematical foundation, classical control theory has widely been applied to generate solutions to robotics control.



(a) Open loop. (b) Feedback loop.

Figure 1: Types of control configurations.

The basic principle of this approach is to devise a method to calculate the appropriate controller (C) that would enable the generation of the proper signals to the plant (P). Open loop control solutions, pictorially represented in Figure 1(a), use the input to directly to determine the current control value while feedback (F), or closed loop control Figure 1(b), utilizes the overall system's performance in addition to the input to determine the control value. Feedback control has been applied in approaches ranging from navigation to manipulation and locomotion [46, 1]. Amongst the advantages provided through the use of a control theory approach are formal processes to design controllers [21] and an established framework to evaluate and characterize performance.

1.2.5 Artificial Intelligence

Many ideas from Artificial Intelligence (AI) have been used in robotics to design robotic controllers. Some of the most prominent ideas include knowledge based systems, neural networks, evolutionary algorithms, search, planning and cognition. As discussed in [3], classical AI methodologies have two key characteristics. The first, abstraction, permits a hierarchical representation of the application at hand that provides a powerful mechanism for problem solving. The second characteristic is that these methodologies use explicit symbolic representational assertion to capture knowledge of the world. Reasoning and planning infrastructures such as STRIPS, ACT-R, SOAR, and OSCAR, search methodologies including A*, and other more modern variants of these all incorporate these two characteristics.

Neural Network (NN) approaches have been used as an effective method to implement real-time robotic control. There have been several works like [10] and [89] which leverage one or more of the many NN forms, sometimes even in combination with other AI techniques.

Whether considering Robonaut [12], robots used for RoboCup [48], or even those used for the DARPA Grand Challenge [96], recent implementations have shown that it is difficult to achieve a capable modern robot without including some AI techniques.

1.2.6 Behavior Based Robotics

Behaviors are real time processes that take input from sensors or other behaviors and send output to effectors [actuators] or other behaviors [52]. Behavior Based Robotics is a paradigm shift from representation and planning, to sensing and acting and in many ways it is a reaction to the approaches of AI. It challenges the notion that knowledge and knowledge representation are critical to intelligence [3].

There are currently several researchers that incorporate Behavior Based Robotics (BBR) approaches into their works, but few would disagree that Arkin, Brooks and

Mataric are among the early pioneers of the field. Among the many advances that these researchers have marshaled is decomposing complex actions into independent behaviors which typically couple sensing and acting closely in a parallel and distributed manner. Such an approach, when possible to attain, significantly reduces the workload required to implement the desired action. BBR has been applied to single and multi-agent robotic systems and one of these pioneers even defined a set basis behaviors [51] as "ubiquitous general building blocks" for composing group behaviors in multi agent systems.

1.2.7 Learning

Many of the approaches discussed thus far have been influenced by biological systems and learning is no exception. In fact, few would argue that the concept of robotic learning is the most biologically inspired as it seeks to incorporate constructs and capabilities into robotics that are unique to humans and other intelligent life.

It must be noted that research has shown that extreme intelligence isn't a requirement for learning and many simple organism and systems of organisms have displayed learning. Most of the examples of this type of learning occur over longer periods of time, larger numbers, and even through the evolutionary adaptation processes.

Considered to some to be a branch of AI, learning has been developed by researchers in many disciplines. One of the pioneers of Machine Learning [5] defines it as the *study of methods for constructing and improving software systems by analyzing examples of their behavior rather than by directly programming them.* This definition makes it clear why learning would be of interest since it can be used to both bootstrap basis behaviors as well as to fine tune existing functionality.

Learning, which is most appropriate where precise specification for the desired behavior is not available but examples of such are, is perfectly suited for a scenario

where instruction is provided by a human operator. Supervised, unsupervised, reinforcement, online and offline learning are all approaches that have been applied to robotics. These concepts will be presented in this section.

Supervised learning is a method where learning is presented with training examples in the form (x, y), where x is the input values and y is the output value. In robotics the input value would be linked to sensor data and the output to action or actuation. The purpose of this approach is to predict the output value y' of a new input value x' which is assumed to be drawn from the same distribution as x. The distribution of x is defined through information acquired from the input values presented during the learning process.

Reinforcement learning is a form of learning that differs from supervised learning. This type of learning uses prototypes of input and a score that indicates the performance of the behavior. This approach accounts for cases where good examples may not be available, and there is a method of identifying what examples are good. The goal of this approach is to maximize some reward. The reward under consideration is not provided to the algorithm, it is discovered by trial and error based on evaluated actions.

Unsupervised learning [81] is another type of learning that seeks to construct feature variables from the observed variables. These variables can comprise either input or output values. This type of learning has proven useful in uncovering patterns in data.

In many applications of learning it is desirable for learning to occur online, where learning is incremental and real-time [92]. There are several advantages that motivate such an approach. Quoting the same researchers,

> online learning is potentially more robust because errors or omissions in the training set can be corrected during operation. Second, training data can often be generated easily and in great quantities when a system is in

> operation, whereas it is usually scarce and precious before ... In a broad sense, online learning is essential if we want to obtain *learning systems* as opposed to merely *learned* ones.

From this brief introduction to learning it can be seen that online learning, supervised learning, and unsupervised learning all provide unique functionality that when appropriately applied can provide mechanism to overcome limitations in some of the techniques presented earlier.

1.3 On Learning from Deterministic Human Action

One possible application of learning is to utilize teleoperation to permit an operator to bootstrap or fine tune a robot's operation. Such an approach would seek to enable a robot to learn from human data and there are some who have concerns about the plausibility and in this section relevant research is presented.

1.3.1 Learning and Determinism

Since it is difficult to learn much from purely random data, it is implied that some deterministic structure(s) exists in human action under consideration. While accounting for noise or local variability, for some, this implication challenges deeply rooted beliefs that determinism is the antithesis of free will [20] and thus cannot be acceptable since human beings innately posses free will [9]. In their view, since humans have free will, then it should not be possible for learning to occur in this manner.

One response to those who raise such an argument is that the humans under consideration are executing a specific task and in most cases will know that they are teaching a (robotic) student agent how to execute the task. If the human was demonstrating a random walk, then the student should only be expected to learn something close to that specific task.

1.3.2 Human Behavior Modeling and Determinism

Human Behavior Modeling (HBM) in military situations [70, 106] is an application that has much in common with teaching. These researchers raise several points of merit, but for this work we cite just two. The first is that human behaviors are variable not just due to simple dichotomies as correctness/incorrectness or experience/inexperience but also because there are multiple possible valid options and because there is variability inter and intra sample of humans (can get tired, may be shorter than others, etc.). The second is that it is possible for a human behavior to be both variable and deterministic if the behavior is conditioned on situational details that are not available to the observer. So the external observer would only be able to account for observed action if differences in internal state were explicitly communicated. Few would argue that if teaching is to occur then this type of communication is required or else the application turns into learning from purely random data. Intentionally limiting variability (as can be controlled) during the teaching process would also be useful so that the student would not get confused.

In addition, research [53] has shown that even flies, insects that appear to act haphazardly, demonstrate deterministic "responses". For these reasons we believe that it should be plausible that learning can happen from human action in the context of teaching and in the constrained domain of teleoperated robotic action.

1.4 Human Robot Interaction

Now that the notion of learning from human action has been introduced, it would be difficult to proceed without addressing Human Robot Interaction (HRI), the field that studies how humans and robots interact and operate. As outlined in section 1.2, robotic control spans from teleoperation to shared control to full autonomy yet lately much of the work done in HRI is focused on making robots act more human-like.

HRI, defined as the field that "regards the study, analysis, design, modeling, implementation, and evaluation of robots for human use" [26], is a broad interdisciplinary field. While there isn't an emphasis on social interaction [13] or on human studies [39], learning from human teleoperation is a valid HRI topic and will contribute much to this body of knowledge.

1.5 Consolidating Approaches

The approaches presented thus far, as varied as they come, do not tend to allow active human interaction once the robot has learned the characteristics of a desired task. For example, once the robot has learned the characteristics of navigation in a particular environment, a new environment in many cases requires reprogramming. Also, if the robot has been trained or programmed with one sensor suite, the addition of additional sensors also requires reprogramming as well. Modification of the task, the environment or the robot thus requires the attention of someone intimately familiar with the inner workings of the robot.

There are some works that display some of the important traits to overcome such challenges. One work, [101], features a humanoid robot that uses real time learning to modify an existing behavior. Human interaction does not feature prominently in this work that uses Linear Weighted Projection Regression (LWPR) to enable learning of sufficiently accurate models of the robot in high dimensional spaces.

There are two works [82, 42] that do use human interaction to facilitate robotic learning. These parallel approaches in effect model the operation of the human, thereby endowing the robot to learn how the human would perform the task. The approach in [82] is composed of two methods, scaffolding and molding, which when combined provide a mechanism that enables new tasks to be learned and incorporated into the robot's capabilities. [42] presents behavior cloning, a method quite similar to molding, which also enabled similar learning capabilities. Both approaches feature

a three part sequence of training, model development, and model execution. These stages were required for each new behavior that is added to the set of behaviors and they were also required for behaviors that needed to be modified. Preprocessing was also required for both approaches, with [42] even generating a grid representation of the world. This method, [42], was devised for a robotic entrant in the Robocup Rescue Robot League (RRRL) and it, like [82], is closely tied to the application and the platform for which it was developed.

To loosen the ties between learning approach and the application/platform to which it is applied, the work presented in [36, 35] seems to be of use. These works demonstrate a method that permits a robot to learn how to categorize relevant or important classes directly from raw data by itself. Such capability proved quite useful in a situation where information was incrementally presented to the robot, and the performance is cited as better than performing the task using Principal Component Analysis (PCA).

An approach that combines the principles applied in [101, 36, 82, 42, 35] and expands upon them, even in a small measure, would be useful and would likely overcome a broad range of challenges. As individual works, most of these are closely tied to the robots they were developed with, and none of them apply the same care to action spaces that they do to the sensing. In these approaches, unsupervised adaptation in sensing, thinking and acting has not yet been demonstrated nor has knowledge transfer been shown which would hint at the ability to capture the essence of the learned behavior.

CHAPTER II

THE INTERACTIVE LEARNING PROCESS

In this chapter several facets of the process of interactive learning will be presented. Where possible, tangible examples of theoretical concepts will be demonstrated. After this chapter, the benefits of interactive learning will become clear, and the tools which provide information about learning will be made evident. All Experiments in this body of research were done with real Amigobot, simulated Khepera robot, and both real and simulated Pioneer 3AT.

It has been long noted in the robotics community that humans are profoundly social species and that social interaction is thus our most natural interaction mode [14]. Socially Interactive Robots (SIRs) are defined by Fong [25] as robots for which "social human-robot interaction is important." They comprise an increasingly active research area. The purpose of this class of robotics is to permit the nature of the interaction between humans and robots to be studied. Ultimately the goal of such research is to create robots that can communicate more naturally.

As an example, if a mobile robot were to share human spaces such as office buildings or homes, traversal of possibly occupied hallways would be a necessary capability. In such a setting, socially interactive robotics would be concerned with enabling the robots to interact in this social setting in a safe, non-threatening, effective manner. Some of these concerns require the robot to understand what the humans in the space are doing. Developing models of human behaviors and leveraging information provided by their programmers, some researchers have been able to create robots that act as receptionists [32] and even deliver items in hospitals.

Like Scholtz [85], many researchers believe that most user interaction with such

robots is still strongly linked to teleoperation. To that end, the term human-robot interaction is often used when researchers are truly referring to human-robot intervention. Intervention is far from the extent of interaction that is possible with robots and other intelligent agents.

When specifically considering robotic learning, it is possible for the robot to observe the provided intervention and its observed properties to generate information. This information could be used to equip the robotic agent to become a more active part of the interaction process. Becoming a more active part of the process makes the process more truly interactive.

Effecting learning in this manner also permits the teacher to engage the robot as one would engage another socially responsive creature. Such function was recognized by Breazeal [14] as a beneficial one in the interaction between human and robot. Social interaction has already been shown as an effective means of interaction with the human in general and can now be leveraged to provide benefit to the HRI domain.

2.1 Learning Interactively from the Beginning

To present the process of interactive learning (IL), first it will be shown how it is possible to apply this type of learning to learn a behavior - starting from zero initial knowledge. In this case zero initial knowledge means that there is no prior information provided about the nature of the behavior. Information exists about the robot, so the sensor and actuator configuration is known a priori. No information is provided however about what values are important or what meaning is to be inferred from any values. Learning is accomplished based on examples of the task provided via teleoperation. Teleoperation is utilized since is removes some of the obstacles typically present when considering operators that may not have a great deal of experience with robotics or programming.

Methods like mixed initiative control [15] and collaborative control [26] both feature approaches where both agent and operator control the robot's hardware. Our implementation of IL takes these approaches a step further, since instruction is provided as needed, and the agent learns from the provided examples. In this manner it is possible to teach a behavior without prior definition of the task. It is our claim that it is possible to apply such a technique to learn to perform the behavior just as well as other more traditional approaches.

2.1.1 Approach

To show that it is possible to apply interactive learning to accomplish learning a new behavior, we first present an overview of the method, then we show how it was applied.

2.1.1.1 Behavior Representation

For this body of work we consider reactive behaviors, those in which actions to be executed are determined solely by current environmental state. These were chosen since they enable significant functionality when coupled with an adequate sensor space. Focusing on reactive tasks also released the need for planning and for learning causal relationships, and since the action is based on information readily available, this also enabled quick system responses. Reactive behaviors can be considered to be Markovian in nature since the next action to be executed is dependent only on the *present* sensory input and not any past values.

Reactive behaviors differ from their more complex analogues, deliberative behaviors, in this singular point. Deliberative behaviors are those that can include information other than present sensory input to determine the current action. The relationship between the current decision and past history is one that can include complex, non-linear linkages. Cognitive reasoning is required to represent such deliberative relationship between sensing and action. This more complex form of behaviors

can be used to capture much broader forms of human intelligence, but there are many useful behaviors that can be captured without its usage.

The reactive behavior can easily be represented as a mapping between sensor data and the actuation data that ought to be executed when it's presented. The relationship between sensing and action is represented as a mapping simply because it is possible (and often the case when considering human teachers) that there is a one-to-many relationship between any given sensory input and the actions.

In addition to a concentration on reactive behaviors, we also consider behaviors which capture specific relationships between the sensory evidence provided and actions executed. Behaviors are thus policies of action that define how the agent operates when specific collections of sensor data are detected. If this relationship between the current sensory input and the action selected is causal, it implies that the behavior under consideration is reactive.

Mathematically a behavior can be represented as a mapping $f : X \rightarrow Y$. In this mapping, X is the sensor space provided to the user for teleoperation, and Y is the action space of the haptic device used to capture human action. Other options exist for the definition of Y. In this body of work those could also include the output issued to the robot, or the state of the robot. The first of these two alternatives is often directly derived from the haptic data provided. However, the second typically requires deep access to the agent's inner machinations. An example of this access includes using a robot's wheel encoders to assess the actual state of the actuator which may be different from the requested state of the device.

Returning to the current representation of a behavior, if the behavior is defined by teleoperation, the mapping between sensing and action is essentially shown by example. A given example (x_k, y_k) indicates that the action $y_k \in Y$ is executed when the sensor data $x_k \in X$ is detected. The example is an "if-then" rule that captures a facet of the behavior.

2.1.1.2 Dimensionality Reduction

For most interesting applications, the mapping between sensing and action is defined over large spaces. It is not uncommon for simple tasks to have a raw representation in more than 10 dimensions and resolutions greater than 256 units along each of these dimensions. Applications which require this much data to represent any relationship make study of the underlying mapping computationally challenging. To mitigate this challenge, dimensionality reduction which preserves the local geometry of the high-dimensional space is often applied.

In our research we are specifically interested in methods which can effectively reduce the dimensionality of the spaces to a single dimension. This results in one dimension for sensing and one dimension for actuation. We are also interested in approaches that could vary the resolution in each dimension in a manner that is somewhat independent of the target behavior. Both linear and non-linear techniques exist which accomplish this task, each with its own merit, but principal components analysis, principal curves, and self-organizing maps are often applied approaches. Kohonen's [45] self organizing map (SOM) is the non-linear technique identified as the most appropriate for this work. We have observed that it permits a more flexible and appropriate representation of the underlying physical system it models.

In our work SOMs, a type of artificial neural network, are used to reduce the dimensionality of the sensor and actuator data. This map, which was initially designed as a data visualization technique, features neurons that are the same dimension as the input to the map. When the input (x) is provided, the neuron (i) that has weights (w) closest (in the Euclidean sense) to the input is selected as the winner. The weights of the nodes in the network are then updated according to the parametric equation presented in (2).

$$w_{new} = w_{old} + \mu h(i)(x - w_{old}) \tag{2}$$

In this equation $h(i)$ is the neighborhood function, which varies inversely with the distance $\|i - w\|$, and μ is the learning rate. As data items are presented, through this update process, the map changes to capture the similarities in the input. Each neuron, has a unique label and represents a distinct element of the input space and this label can be used as the encoded state.

In this work, separate SOMs were used to process sensing and acting data associated with a desired task scenario during teleoperation. The sensing data was comprised of all the sensor values that were observed by the robot at a given measurement opportunity. Analogously, the acting data was comprised of all the actuator values issued to the robot at that time.

As the maps self organize, this space represents the actual range of observed sensor or actuator data. When new sensor or actuator values were generated, these and the preceding values of that type are also presented to the map sequentially from oldest to newest. In this manner, the maps reflect all the data presented to the robot, and are compounded over the experience of the robot. Varying the number of neurons used (the number of classes provided) would not change any of these properties, however it can potentially affect the manifold's ability to represent the data with enough detail.

To present a tangible example of how this process operates Figure 2 is presented. This figure shows the SOM that is generated after being trained with actuation examples demonstrated during a navigation task. This task utilized two actuators and this map captures the manifold which shows the actuator values actually relevant to the task. This manifold does not cover all of the pairs of values possible (real, two axes space quantized to two decimal places). In this case the (100) nodes present capture pairs of actuator values which represent degrees of left and right turns, as well as maximum forward velocity.

Since the manifold is designed to best cover the presented examples (i.e. the provided instruction), it does not cover actions that were never demonstrated. Further,

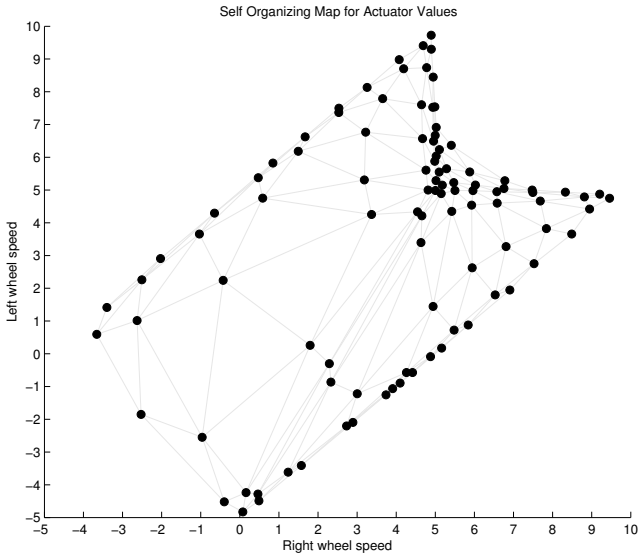


Figure 2: Self Organizing Map of actuator values after some training. The embedded manifold clearly shows that there is underlying structure in the examples presented. This structure can be more efficiently captured in a lower dimensional space.

the manifold is adaptively quantized to provide the highest resolution over the space where finest granularity is needed. This feature is one of the most useful aspects of applying this technique to reduce dimensionality in this system. Adaptively tuning the resolution to cover the spaces most critical to successfully learning permits previously unknown behaviors to be taught. This feature also permits the number of states needed for either sensing or actuation to be selected without any prior knowledge of the task.

Sometimes it may be useful to have more states to cover the considered space. The application of the Growing Hierarchical SOM [56], or some other growing grid method [28], can help in such a situation. In any case, if there were not enough states, the result of the learning process is akin to learning from poor instruction; There will be large error values as learning progresses since the map is not able to better map the space over time.

If there are too many states a different challenge is presented. In this case it is

possible that the map can cover the required space without changing the location of its nodes much. The benefits of having high resolution where it is needed most and low resolution where it is not needed are not provided. Further, more information than really needed is used to represent the learned behaviors and this reduces the value of useful pieces of instruction. This value will be shown in subsequent chapters to be a useful tool for the student to assess its own performance.

2.1.1.3 Behavior Representation in a New Space

By applying dimensionality reduction in this manner to the sensing and actuation spaces, the mathematical formulation of this problem is changed. No longer is a behavior represented as a mapping between sensing and actuation, but it is now a mapping between sensing and actuation states. A behavior is now represented as a mapping $\hat{f} : \hat{X} \rightarrow \hat{Y}$. Where the transformations $X \rightarrow \hat{X}$ and $Y \rightarrow \hat{Y}$, are produced by dimensionality reduction.

At this point, any approach can still be applied to learn the mapping $\hat{f}$. The use of the SOM to effect dimensionality reduction preserves raw data (sensor or actuator) used in the weights of each of the nodes (referred to as a codebook in collective form). This data could then replace the original data set, thereby reducing the quantity of data stored if an instruction history is being recorded. The result of this step is that from the provided example set (3), the mapping between sensing and action shown in (4) can be generated.

$$B = \{(x_k, y_k)\}, \quad k \in \{1, ..., N\} \tag{3}$$

where N is the number of training points provided.

$$\hat{f} = \{(\hat{x_l}, \hat{y_p})\}, \quad l \in \{1, ..., L\}, p \in \{1, ..., P\}, \tag{4}$$

where L and P are the number of classification points provided for sensing and

actuation respectively.

2.1.1.4 Learning Interactively

For the reasons previously outlined, the source of instruction applied in this approach is an external teacher. In many cases this teacher will be a human operator, however interactive learning (IL) can be demonstrated by non-human instructors. The teacher's role is to provide examples of what they consider to be correct function, in the process of demonstrating the behavior. While this is a very subjective concept, the teacher is attempting to provide examples of a behavior to a student. This means that the teacher should have an understanding of that behavior, and is trying to get the robotic student to learn a specific behavior. The assessment of whether the behavior is being demonstrated is also provided by the same teacher, so it is expected that the subjective assessment will be consistent in both of these cases. In this implementation of IL, examples are provided via teleoperation, and as indicated, they are provided as needed by the teacher.

The teacher provides examples of what is assumed to be "correct function" of the target behavior. Their examples are the only source of insight for the agent to learn the task. As can be expected, the better the examples the better the learning process ought to progress.

The examples are also provided via teleoperation, so there is no translation required for the agent to apply the provided instruction. This does pose some challenges in terms of situational awareness [22] which can have a negative impact on the process. This challenge is not a focus of this work. We will assume that the teacher is able to perform the task at an adequate degree, but if not then their instruction will be equivalent to other cases where useless instruction is provided. The instruction is also assumed to be provided only as needed. This expectation utilizes the judgment of the instructor, the human in the loop who is also providing instruction.

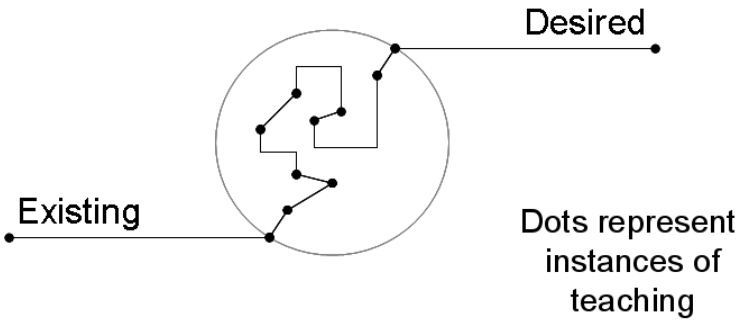


Figure 3: Representation of the effect of instruction on the student's performance in a favorable case.

Once examples are provided, they are used to update the agents existing capabilities. This incremental addition of information should have the resulting effect of improving the performance of the task. Applying the update technique described in the previous section, new instruction is used to update the agents' ability to perceive what action state was demonstrated in response to the provided sensor state. By updating the SOMs used to adaptively map the sensor and action spaces into sensor and actuation states and then re-interpreting prior instruction, it is possible to capture information from all of the learning process.

The approach described thus far can be applied successfully in a non-interactive manner. What makes this implementation interactive is that as soon as examples can be incorporated, the agent will attempt to demonstrate what it knows. The teacher permits the agent to do so and only steps in to provide new examples (instruction) if they perceive that the agent's actions warrant intervention. In this approach, the agent communicates by examples what it can accomplish. The teacher watches to see if the student is "failing" or performing poorly at the task.

The intended interaction between robotic student and human teacher can be captured pictorially in Figure 3. Before entering the "circle of interaction", the robot has some existing capability (which can be completely inadequate). Through the instances of teaching provided by the instructor, it is expected that its performance will

change. It is the instructor's goal to provide additional instruction until the desired performance level is consistently demonstrated. Once the desired performance level is attained interaction between student and teacher ceases.

This process is very similar to scaffolding [105], in which teachers provide support while the student learns to master portions of a task. Gradually the supports are removed as the student attains greater skill. The teacher determines what supports are needed and when they should be removed based on their general experience teaching. The teacher's specific experience interacting with the current student also affects their decisions. A common example of a scaffold is the use of training wheels when a young child is learning to ride a bicycle. In the case of this research, the scaffold is the actual set of examples provided to the student. When the student has mastered certain portions of the desired task (i.e. can demonstrate the desired performance level), the teacher no longer provides those kinds of examples.

2.1.1.5 Accessing Prior Knowledge

To conclude the presentation of the interactive learning approach, the process used to tap into the previously acquired knowledge is presented. As presented in Section 2.1.1.3, the mapping $\hat{f}$ is used to capture the observed relationship between sensing and action. Through the use of the SOM to effect dimensionality reduction, the action state $\hat{y}$ contains the encoded actuator values, thus these values can be directly extracted once the actuator state is identified.

To identify the actuation state the algorithm presented in Algorithm 1 is applied. This approach is naive in that it does not assume it can interpolate between provided examples of sensing or actuation. If the current sensory input is not recognized, the algorithm does not produce a result. This conservative approach is implemented since the differences between neighboring sensor or actuation states is a function of their appearance in the provided instruction sets and is not directly related to value

Algorithm 1 Process of identifying actuation state

```
Current sensor values converted to sensor state
if sensor state recognized in knowledge base then
  if Multiple associated action states found then
    Population generated with distribution akin to provided instruction
    Winner selected randomly from population
  else
    Winner = action state
  end if
end if
```

judgments about the meaning of similar things. While the use of classifiers appears to perform exactly that value judgment, it does not. The classifiers are used to effect perception. They permit the system to generate action and sensor states (percepts), and they permit the system to generate a notion of states which are different, but they are not used to determine states which should exist.

This approach can be described as an implementation of a probabilistic look-up table. It is fast, easy to update, and also enables exploitation of prior knowledge and exploration of the space of instruction. This last advantage, the coupled exploration and exploitation is quite useful since it is favorable strategy when considering learning from examples in this manner.

This method can be implemented in near real time so that it can be naturally interactive. In most cases the system operates faster than 8 Hz with a 300ms delay. When implemented in hardware, with wireless connections between the robot and the instructor, this delay can rise at the whim of the TCP/IP transport layer. On occasion, delays up to 2 seconds were observed when using 802.11g router. Once the robot received intial instruction, the issues caused by such a delay were mitigated, even with a small number of examples of the target task.

2.1.2 Experiments

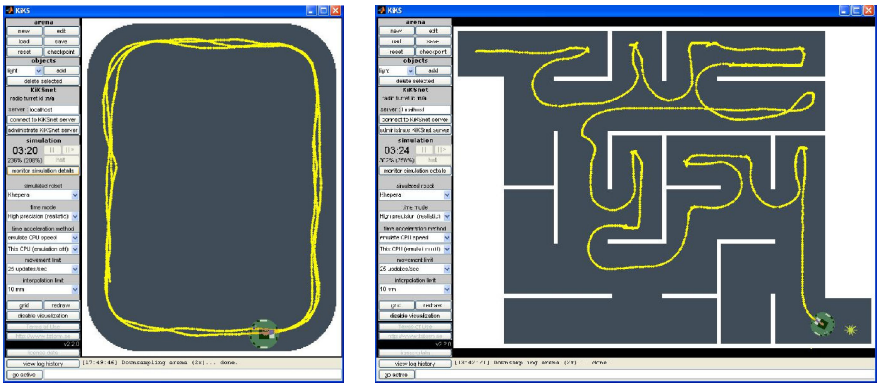
To present scenarios where learning was attained, the following experiments are considered. The experimental goal was to assess the algorithm's performance in simulation and on hardware platforms in both simple and complex environments. The behavior selected was wall following, a standard element in many mobile robotic applications. In each of the experiments, the teacher, after some practice performing the task with the robotic platform, used a joystick to teach the robot. Unless otherwise stated, the maps used for sensor and actuation data were each populated with one hundred neurons spread over a ten by ten grid.

2.1.2.1 2D Simulation Environment

The first experiments are performed with a simulated Khepera robot. This simulation was provided by the KiKS [64] project. This MATLAB based simulator implements a 2D discrete event simulator and provides access to a simulated robot and many of its sensors - most notably it's sonar and ambient light sensors. In the arenas presented in Figures 4(a) and 4(b), the target task was to navigate in a specific manner without hitting walls. In the maze environment, the target behavior was to follow the wall to the left side of the robot until it passed through the goal location (the yellow light). The robots only sensors for these tasks are infrared proximity sensors. Ambient light sensors are used to determine the conclusion of the trial (proximity to light), but these sensors are used only for the supervisory code which manages the robot as it learns.

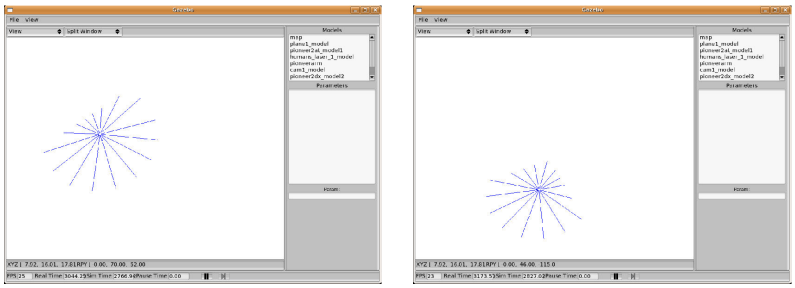
2.1.2.2 3D Simulation Environment

In the next experiment we consider a simulated robot using the Gazebo 3D simulator [30]. This robot, equipped with 16 laser proximity sensors, was taught a similar navigation task (See Figure 5). An arena like the one in Figure 4(a) was created and instruction provided interactively until the task was learned. In this environment,



(a) Simple environment. (b) Complex environment.

Figure 4: Overhead view of simulated Khepera robots.



(a) Robot traveling along wall. (b) Robot near a corner.

Figure 5: Overhead view of simulated Pioneer 3AT robots. The blue lines are the rays projected from the laser sensors. Simulation in Gazebo.

these experiments were repeated two more times, first with only vision data (images) from an onboard camera, then with vision data and data from the laser sensors. Figure 6 shows the view from one of the onboard cameras used in this experiment.

2.1.2.3 Hardware Implementation

Finally, experiments were done with an Amigobot robot. In these experiments, the goal was to teach the robot to navigate safely in the environments shown in Figure 7(a). The robot was taught in each of the pictured environments beginning from zero initial knowledge. Eight sonar sensors and an Axis 209 wireless camera were used to provide sensor input for this task.

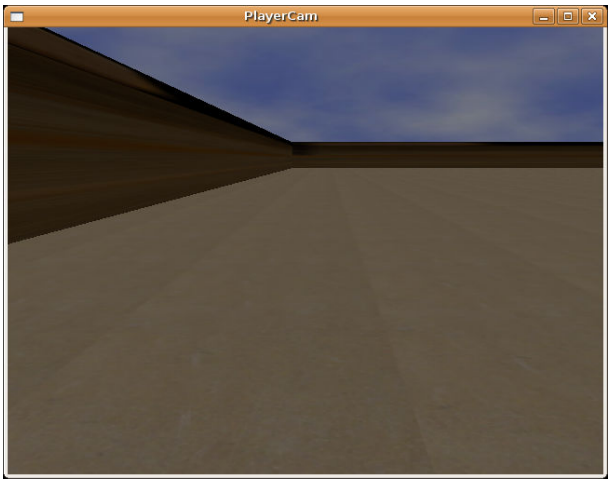


Figure 6: On board view from simulated Pioneer 3AT robot. Simulation in Gazebo.

2.1.3 Results

To show that indeed it was possible to apply interactive learning to learn a previously unknown task the following images are presented. These images show the paths that robots were able to demonstrate after the interactive learning process. The paths are presented for varying durations of interaction.

2.1.3.1 2D Simulation Environment

The first set of paths are presented for the simulated agents provided via the MATLAB based KiKS simulator. This robotic agent's instruction was based on infra red proximity sensor data and differential drive actuation. The paths of the robot shown in Figure 8 show the progress of learning over time. In these paths show the improvement in performance after learning from 10, 25, 150, and 400 examples.

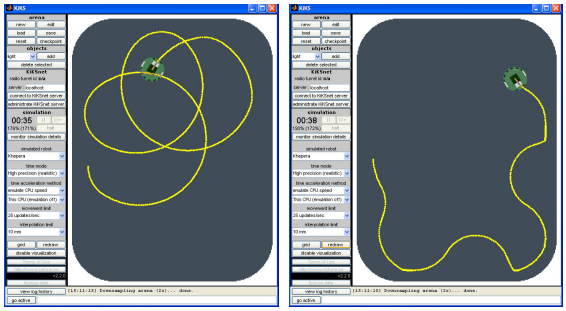
This graphical representation of the effects of learning is just one form of confirmation of the hypothesis of this section. Method such as dynamic time warping and other approaches which seek to quantify the performance by the application of a metric will be introduced later in this thesis.

(a) Real Amigobot (inset) in environment where training was performed.

(b) An environment different from training environment.

(c) An environment with varied textures surfaces.

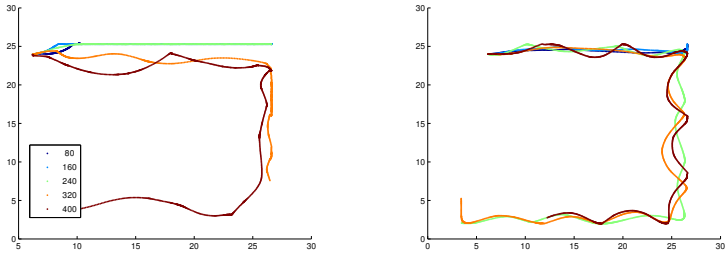
Figure 7: Hardware form of the Amigobot and the environments which it was taught and its performance tested.



(a) 10 (b) 25

(c) 150 (d) 400

Figure 8: A single behavior learned as more human examples are provided. Each subfigure shows the path demonstrated by the simulated khepera robot at various stages in the learning process.



(a) Laser data only. (b) Camera data only.

Figure 9: Paths demonstrated for behaviors learned with single sensor modalities. The colors indicate the behavior indicated after learning from 80, 160, 240, 320, and 400 examples.

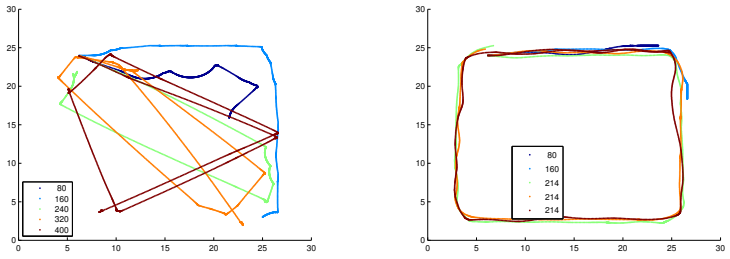
2.1.3.2 3D Environment with Single Sensing Modality

When considering robots that received instruction received from a more complex three dimensional environments, similar stories are presented. These figures also confirm that learning was attained. In the simplest of these cases, that presented in Figure 9(a), the sensor values were limited to the robots' laser sensor. These sensors capture a two dimensional representation of the environment by encoding the distance between the sensors and the closest detectable obstacle. There is a natural transition between the two dimensional environment first presented and this case where a sensor is utilized which provides an analogous view of the world. It is good that the agent was able to perform similarly in these cases.

When the sensor space was changed to include data based on vision, it is encouraging to note that the same confirmation is provided. In this case were an order of magnitude more data is consumed during the learning, the robot is able to effectively benefit from the provided instruction. Further, vision is provided from two forward facing camera sensors, one connected to the chassis, and the other to the "wrist" of the robot's actuated arm. The cameras have limited fields of view (60 degrees) and thus at times the wall, the location of which was quite relevant to the tasks taught, was not in view. This shortcoming is one of the major reasons there is a wobbling artifact in the paths presented in Figure 9(b).

Comparing figures 9(a) and 9(b) it can be seen that the robot was able to perform the task, in spite of this limitation of field of view. It can also be seen that the robot is able to learn to perform the task more quickly as in each of these cases the same number of examples is used to plot paths in the same color.

It should be noted that the performance did not improve linearly as examples were added. In some cases the performance decreased. This occurrence indicates that it may be the case that more examples do not always improve the performance - a theme that is addressed in Chapter 3.



(a) Wall Avoidance. (b) Wall Following.

Figure 10: Paths demonstrated for two behaviors learned with both image and laser proximity data.

2.1.3.3 3D Environment with Vision and Proximity Sensing

Providing both vision and proximity modalities performance was better than previously seen (i.e. stayed closer to the wall than before) however it can be seen that just as in the case with vision only, performance did degrade at the end. The negative effects of the limited field of view were clearly compensated for by the addition of the proximity sensors. This point is made clear since the wobbling artifact in the robot's path is no longer evident.

2.1.3.4 Hardware Implementation

The transition to client code which interfaced to the player server on the Amigobot was a successful one. In this case, the robot was able to navigate the rooms considered. Sonar sensors were used on the Amigobot and the sizable noise present in these sensors did have an effect on the robot's learning. In addition to the wobbling artifact, attributed to the limited field of view of the camera, there were deviations from the type of paths shown in Figure 10(b). These deviation were attributed to the use of sonar.

2.1.4 Summary

At the beginning of this section a claim was made that it was possible to apply interactive learning to learn a behavior with no a priori information about the behavior.

The results presented show examples where this was indeed shown. These results do not express support for this claim in a quantitative manner, but they do show that the robotic student was able to learn to perform the task. Qualitative treatment of the interactive learning process is to be covered in the following section. Specifically the process will be compared with well established techniques used to endow a robot with new skills.

At the conclusion of the sections in this chapter, the reader is expected to have a solid grasp of the concepts applied in this implementation of interactive learning. This section has described the process and in the following we continue by showing how it compares with other methods. What should be attained after reading the following section is a better grasp of how the mechanics of interactive learning tangibly change the student, thus showing some of the strengths of this approach. There is no claim that interactive learning is the only, or even the best approach to learning a behavior from zero initial knowledge. There is however an assertion that the process of interaction is key in equipping robots to function in the future.

2.2 Validation of Learning Interactively

The purpose of this section is to investigate the hypothesis that interactive learning is beneficial and determine whether its performance is competitive with more established methods. Unlike in Section 2.1.1.4 where interactive learning was first introduced, in this section, a more quantitative approach is presented Other researchers have also made advances in this niche and have shown great results to that end [35, 18]. Later in this work we present how properties of the interaction can be leveraged to provide additional benefits, benefits which are - at least in our assessment - much more useful to this application in the service robotics community.

Before this treatment continues, an important note must be made. It is very difficult to have an unbiased "apples to apples" when considering interactive learning

methods. The interaction process is an essential component of the presented learning process and by definition, strict repeatability is not possible. The learning process is directly related to the experience gained by the student over the course of the interaction process as well as the mindset of the teacher during the process. It is quite possible that the same teacher could have a different learning outcome (possible better or worse) if their instruction varies. It is also possible that the same teacher could have a different learning outcome if the environment was different. Finally it is possible that a student could have a positive learning experience if they selectively ignore portions of the instruction it receives.

All of these possibilities can easily be observed in scenarios where human teachers and students interact and in this work it is assumed (asserted) that the same also applies to robotic students. These factors all make the learning process challenging and their presence provide good reasons for skepticism about the hypothesis under current investigation.

2.2.1 Learning Approach

A basic definition that we employ is that a robot is said to have learned if it can perform a specific task or attain a specific goal that it was not initially designed to perform, and it is able to do so only after some identifiable process occurred that changes how it operates. This identifiable process can take many forms; some of which will be discussed further, but they all can be generally classified under the title of teaching. Whether programmed with the ability to self teach, or the teaching is provided by an external source, the approach taken to teach is inextricably linked to the approach taken to learn. If the teaching is not performed in the proper manner learning, especially in the constrained robotic application/sense/domain, cannot occur. For this reason, teaching and learning are often used interchangeably - although teaching focuses on the (human) teacher and learning on the robotic student.

This definition of learning steers clear of the debate on Emergent Behaviors [37]. It is duly noted that surprising or unexpected behaviors can arise out of the "learning" process but in this work emphasis is placed on behaviors that are desired or specifically intended to arise out the interaction process. While robots are not yet able to demonstrate all the capabilities of human learning, the scope of this research is limited to one of of these - the ability to acquire specific skills. Equipped with this understanding of learning and teaching, we now return to the concept of interactive learning.

2.2.1.1 Interactive learning

The pedagogical roots of IL lie in the teacher sharing relevant information with the student and the student testing the limits of their understanding of what they have been taught. Through this process, the teacher imparts new knowledge and when needed, supplements existing knowledge to correct misunderstandings and fill gaps. Effective teachers use a technique called scaffolding [105, 13], to enable the student to test the limits of their understanding in a controlled learning space. Scaffolding, and IL in general, requires that both teacher and student learn and adapt to each other simultaneously, a key difference between these and other learning approaches.

The benefits of such an approach are linked to leveraging the strengths of human and robot at the same time. Few would challenge the statement that humans are intelligent, so in the case where a human expert is present it would make sense to incorporating their knowledge to solve a problem. Further, robots endowed with their computational power, can be utilized for data analysis and processing just as computers have done for some time. Both robots and humans have their strengths, and IL can enable both types of strength to be combined to solve problems and to improve the robot's ability to function in the future.

A useful quality of IL is that it allows exploration of the learning space. This

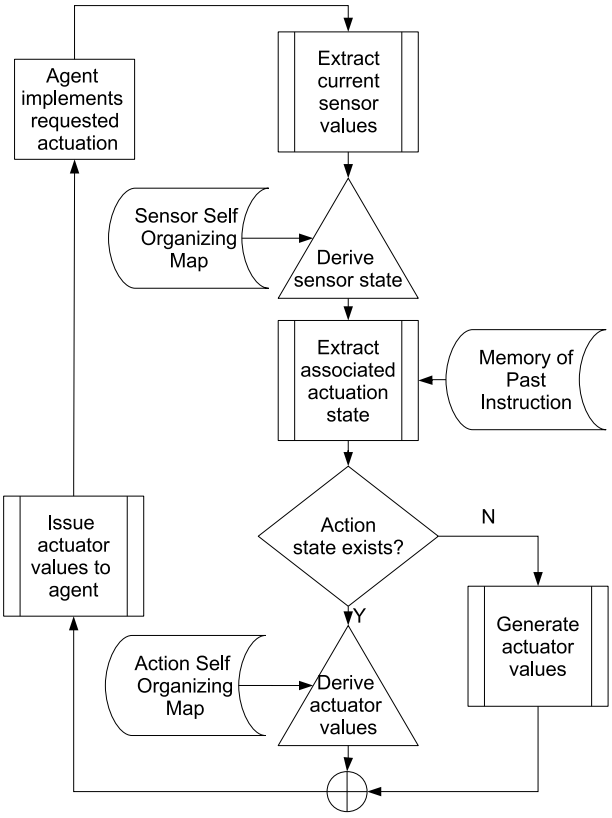
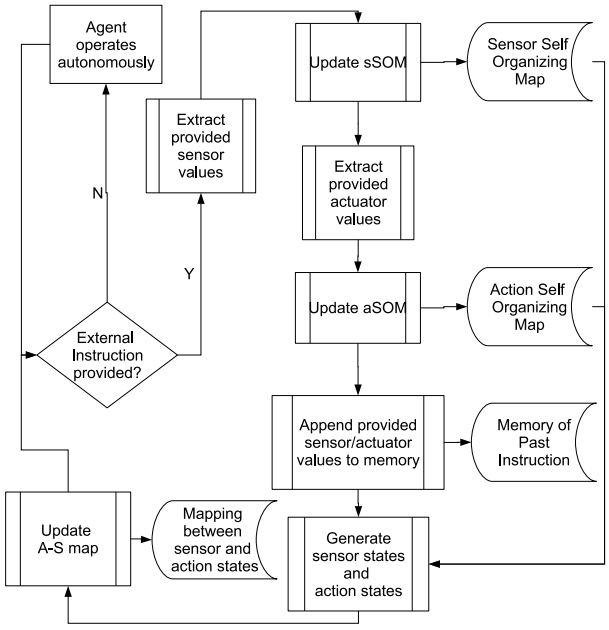


Figure 11: Flowchart describing the autonomous portion of the agent's operation.

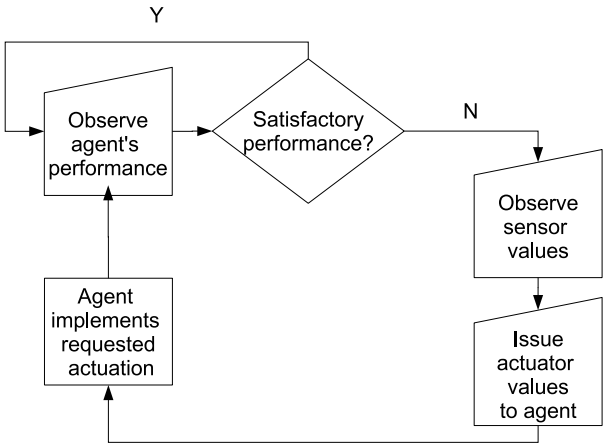
enables more of the problem space to be presented to the robot and allows not only the desired task to be presented, but also the components required to return to the desired task after some deviation. This quality provides the basis for coping with noise and other sources of uncertainty.

The flow chart in Figure 11 shows the process through which the student applies the acquired instruction to demonstrate its capabilities. In this application of interactive learning, when the student does not receive instruction, the previous experience is utilized. This reliance on past instructions permits the student to function. When instruction is provided, the past memory and the perceptual filters provided by the SOMs are updated. After this update process, or in threaded implementations concurrently, the instruction is also passed on to be implemented. This relationship is captured in the flow chart presented in Figure 12.

There is one more benefit that we actually show in this work, namely performance. Bentivenga's work [8] shows that there was a performance improvement greater than 25% when a robot is allowed to observe extra trials prior to beginning a period of self training. Howard [39] also shows that such a benefit is achieved by human operators as well. While these works did not involve online interaction, they indicate that there is something to this notion of interactive learning.



(a) How learning is incorporated into the agent.



(b) How instruction is provided to the agent.

Figure 12: Flow charts describing the process of interactive learning.

2.2.2 Standard Approaches

2.2.2.1 Neurocontroller

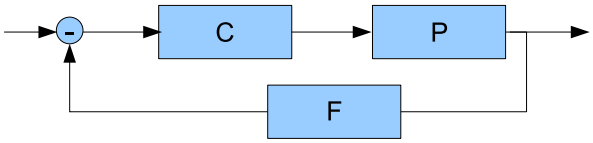


Figure 13: Feedback control loop. In this figure C is the controller, P is the plant (for this document, the robot), and F the feedback path. In most cases the feedback path is provided through the robot as perception information which is a product of the sensor data the robot acquires.

For purposes of comparison, two techniques were implemented to permit a robot to implement/demonstrate a target behavior. They both feature different methods to generate the controller, C, in Figure 13.

The first, a neurocontroller, was patterned after the implemented works associated with [64]. In that work the neurocontroller was implemented using a single input layer composed of a node for each sensor input and an output layer with a node for each actuator value. A representation of this controller can be seen in Figure 14. The weights provided for the target behavior are shown in Tab. 1. These weights were tuned to perform the task with the Khepera robot.The actuator value, a_i, is defined by the equation presented in 5. In this equation, W_{ij} is the weight for the edge between actuator node i and sensor node j. Finally, s_j is the value of the j^{th} sensor.

$$a_i = \sum W_{ij} * s_j \tag{5}$$

Table 1: Weights W_{ij} used for the neurocontroller.

i	j							
	1	2	3	4	5	6	7	8
1	-1	4	0	3	0	0	0	0
2	3	0	0	-8	0	0	0	0

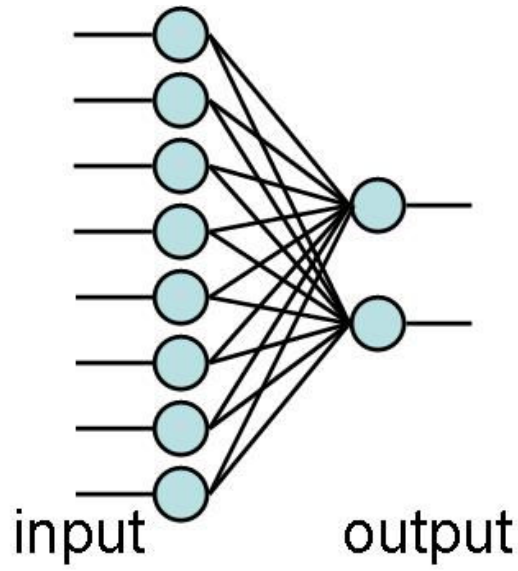


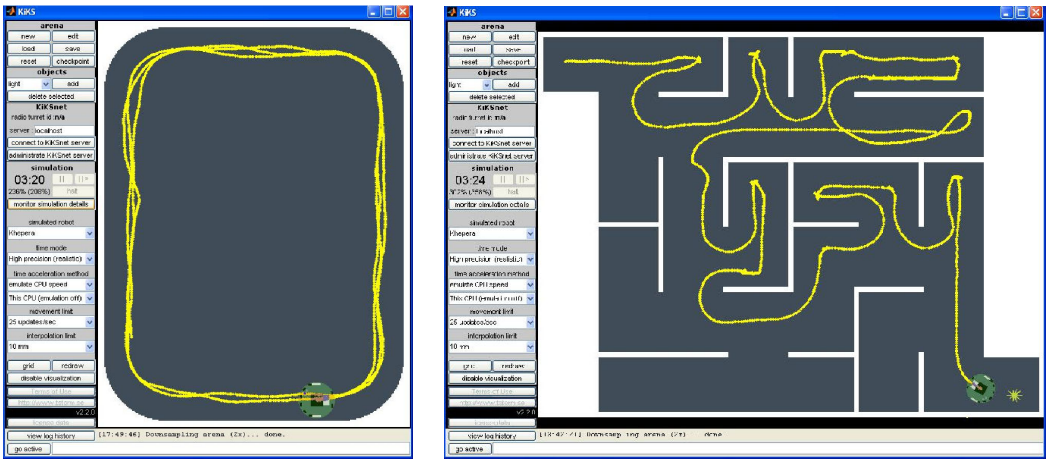
Figure 14: Graphical representation of the structure of the neurocontroller. For this network no bias term is used thus no additive term is required.

2.2.2.2 PID controller

The second method, a PID controller, was implemented in a slightly different manner. The velocity of the robot was decoupled into two parts: translational and rotational.An error signal was defined to be the difference between the current rotational velocity and the required value. To generate the new value for the rotational velocity, the error was combined with its integral and derivative with respect to time. Each of the three terms was scaled by constants K_p, K_i and K_d as shown in Equation 6. For this work $K_p = 6.5 \times 10^{-5}$, $K_i = 0$ and $K_d = 5.4 \times 10^{-5}$. The translational velocity was treated separately and varied proportionally with the distance between the robot and the nearest obstacle in the direction of motion.

$$Rot = K_p * error + K_i \int error\ dt + K_d \frac{d}{dt}\ error \tag{6}$$

The actuator values were generated by combining translational and rotational velocities and then these were passed onto the robot.



(a) Simple environment. (b) Complex environment.

Figure 15: Simulated Khepera in different environments. The yellow dots indicate past locations of the robot as it traversed the arena.

2.2.3 Experiments

To explore the hypothesis covered in this section, experiments are presented for the most controlled case - i.e. that where the robotic agent is simulated in the discrete event simulator KiKS [64].

2.2.3.1 Target behavior

The experiments in this work used human subjects with varying degrees of expertise to teach a mobile robot interactively. Each subject performed the target behavior teleoperatively for one lap then entered into an interactive learning phase where the robot learned for ten laps. The test controllers were both derived a priori and observed to perform the tasks as desired.

The arenas shown in Figure 15 depicts the environments that the simulated Khepera robots were exposed to during these experiments. The environment shown in Figure 15(b) provided the robot with more complex sensory stimulus, but the same reactive behavior was demonstrated by the teacher in both environments.

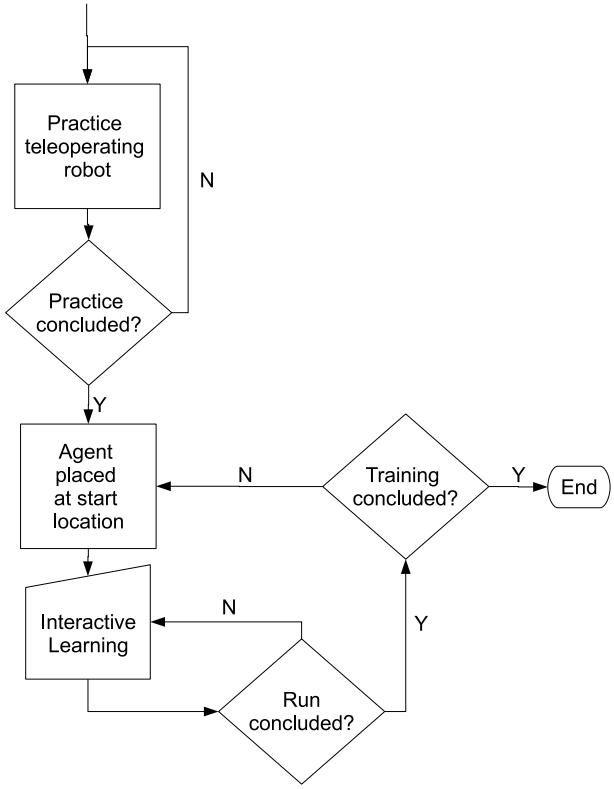


Figure 16: Flow chart describing the process used to collect data.

2.2.3.2 Capturing Instruction

During the process of interactive learning, as outlined in Figure 16, the teacher provided examples of the target behavior incrementally as they observed the robot's actions. Two axes of a multi axis joystick were used to capture human action, and instead of providing just sensor values to the human, the overhead view of the robot in the simulated arena was provided (see Figure 15). Whenever examples were provided to the robot it incorporated them into its behavior and then demonstrated the updated behavior. Through this incremental process the user and the robot simultaneously adapted to each other.

2.2.3.3 Evaluating Performance

To evaluate the performance of the wall following behavior, the examples gathered during interaction were grouped into laps. To calculate the performance after the

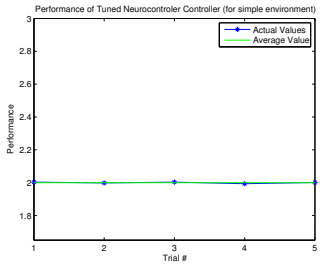
k^{th} lap, the examples provided during laps $1, ..., k$ are collected and used for learning. This is one advantage of the learning approach used in that it is able to apply data interactively or in batch form.

The metric used to compute performance is presented in (7). It is a two part construct of the time to complete a lap, t and the distance to the nearest wall, d. The weights α_1 and α_2 are extracted from the average values of $1/d$ and $1/t$ when the behavior was executed using the tuned neurocontroller.

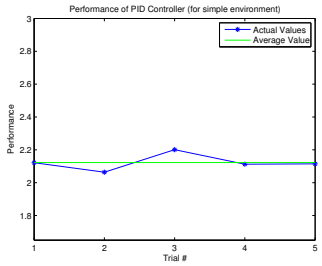
$$\text{performance} = \alpha_1 d + \alpha_2 t \tag{7}$$

By definition the performance of this controller is 2 and for the listed performance metric, *smaller* values are associated with improved performance. It is noted that expert human operators can demonstrate better performance than this coded controller, but the purpose of using its weights in this manner is to provide a basis for comparison.

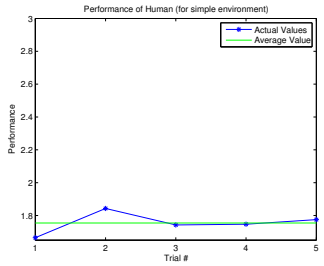
Dynamic time warping (DTW) [43, 66] was also considered to generate a performance metric. With this approach, the difference between the reference path and the current path are evaluated by accessing a non zero cost for any incremental step which differs for the reference sequence. The closest approximation of the reference path is generated from the incremental steps in the evaluated sequence. The cost along this path is then summed and used to provide the performance metric for this path. This multistage process is quite time consuming for long sequences and does not lend itself to incrementally accessing performance. The results of unpublished indicate that the application of DTW provided no benefit over the previous performance metric so it will not be utilized moving forward.



(a) Performance of neurocontroller.



(b) Performance of PID controller.

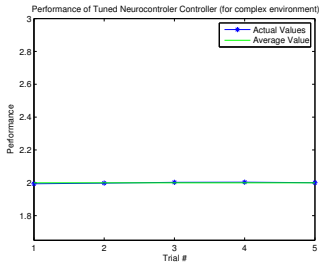


(c) Performance of human teleoperation.

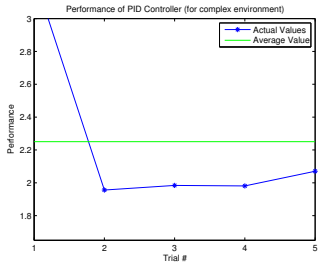
Figure 17: Data gathered in simple environment.

2.2.4 Results

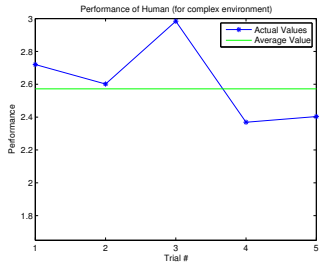
The performance of the neurocontroller, PID, and the human teacher operating in the simple environment is shown in Figure 17. These figures confirm that the human can perform the task well, and that on average the tuned neurocontroller performs better than the PID.



(a) Performance of neurocontroller.



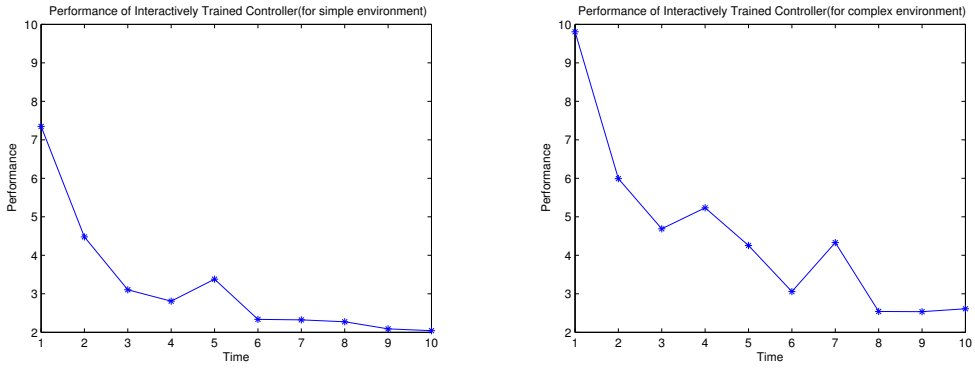
(b) Performance of PID controller.



(c) Performance of human teleoperation.

Figure 18: Data gathered in complex environment.

A similar story is told in the complex environment with one deviation. In this case the performance of the PID is closer to that of the neurocontroller but the human does not outperform the other controllers. The human's performance in the complex environment as measured by this metric indicates that there is a moderate performance advantage when automation is introduced. This can be seen in Figure 18. It is useful to note that the neurocontroller performed consistently in both the simple and the complex environments as would be expected since it was tuned to



(a) Simple environment.

(b) Complex environment.

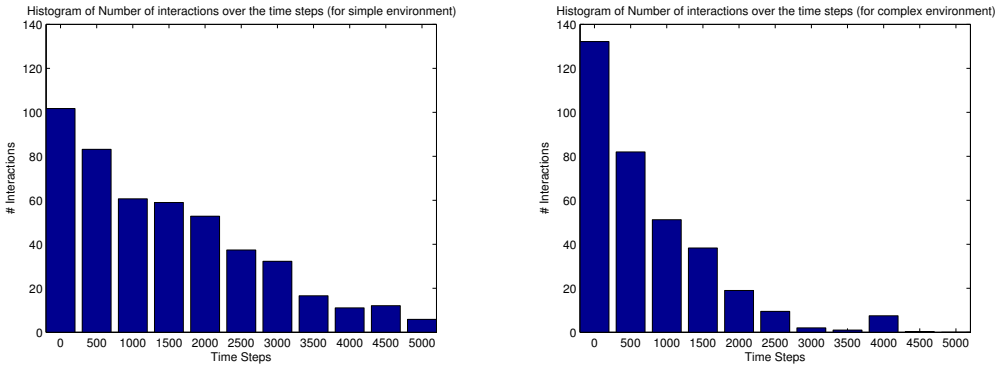
Figure 19: Performance vs. interaction time for two simulated robots which have learned to perform a navigation task in different environments.

perform this task.

In Figure 19, the performance curves for interactive learning in both environments are presented. These curves, which are generated by evaluating the robot's performance with additional instruction, show that the robotic student's performance improved over the period of interaction. After ten laps, in each case, the performance was very close to the performance values measured for human teachers in each environment (1.75 and 2.54). These curves capture the trend of performance with additional instruction and will be discussed in greater detail in Section 2.3.

Figure 15, which was introduced earlier, shows the paths of simulated Khepera robots which learned interactively in simple and complex environments. The images shown in this figure confirm that the robot was indeed able to learn how to wall follow in each of these environments. The performance was displayed after a single interactive learning bout (ten laps) with the teacher. Figure 20 shows how the number of interactions vary on average during a training bout.

The number of interactions required when the human aided the robot via interactive learning was an order of magnitude smaller than when the human performed the task in the simple environment alone. When in the complex environment to



(a) Simple environment.

(b) Complex environment.

Figure 20: Histogram of number of interactions over interaction time.

accomplish the desired task the human interacted four times more without the benefit of interactive learning. This difference will shows that there is an advantage to interactive learning when concerned with reducing the load on a human operator.

2.2.5 Summary

In this section we have confirmed the qualitative evidence about the manner in which learning affects performance with quantitative measures. These measures show that performance did improve in time and that the performance approaches that of more traditional approaches to skill transfer. More detail was provided about the implemented interactive approach and with this understanding further treatment of the usefulness of the interaction can be considered.

2.3 The Robotic Learning Curve

With the previously acquired understanding of the acquired implementation of interactive learning, and exposed to the process of measuring performance, we now consider how the performance varies with time. Characterizing this property will provide a great deal of insight about learning interactively. In this section, the learning curves generated for interaction between an expert user and a learning mobile robot are presented. Interactive learning [74] and learning from teleoperation were

performed and the curves extracted for both types of learning are presented.

2.3.1 Families of Learning Curves

Several methods have been applied to evaluate the process of human learning. In spite of this, as early as the turn of the century [54] it was noted that there are strong regularities in the results obtained from these methods. One area where these regularities are evident is the *learning curve*. The learning curve is a representation of the learning assessment (performance) as the learning process occurs.

As noted by [54, 78] and several others, learning curves for human subjects are generally described by two families of functions: exponential functions and power-law functions. The exponential family of functions is defined by (8). This equation shows that the performance, P, is dependent on N, the number of practice instances or trials. To accommodate for cases where the learning process is applied and prior learning exists, N_0 is the number of initial instances or trials. $N + N_0$ thus represents the total number of instances or trials. P is also defined by constants A, B, and β. These constants respectively capture the steady state performance level, the range of learning, and the learning rate of the student.

$$P(N) = A + Be^{\beta(N+N_0)} \tag{8}$$

The power-law family of functions is defined in (9) and all the parameters and variables have analogous meanings as those presented in (8).

$$P(N) = A + B(N + N_0)^{\beta} \tag{9}$$

In general learning curves capture the performance of a task over the course of the learning process. It is true that for many human subjects, past experience will influence the learning process, but these parametric forms of the learning curve were developed based on the number of times direct training occurred. The value N_0 refers to prior training, but since it is difficult to effectively capture all types of prior training for human students, these curves are derived from known instances of training.

For this reason, common practice sets $N_0 = 0$, then applies a regression method to derive the values B and β. Although this is the most accurate form of (8) and (9), it is commonly seen in the literature that the equations are further simplified by setting $A = 0$.

In terms of identifying which of these two families of functions should be appropriate, [39] indicates that Equation 8 should be applied when fitting performance of a single user exploring a single strategy. This will not be the case in the situation when teachers are interactively teaching students. The relevance of such a scenario will become more obvious. Briefly, in this scenario, teachers typically adjust their instruction based on observed or apparent learning progress thus creating a system in which the learning attained is directly related to the proportion of what remains to be learned. Such functionality lends itself to the theory of learning curves which follow the power law [78].

When available, learning curves can help to provide a wealth of information about the learning process. These curves can be used to estimate how long or how many instances it will take to display a desired performance level. They can also be used to estimate the performance level after a given amount of time (instances) passes. Such information can be combined to identify at what point the learning process should be terminated since the increases in performance level no longer attain significant changes. The merits of having this information have been presented in [75]. Of specific note is that each learning curve also provides the learning rate, β, which in itself gives great indication of the quality of the learning process.

It must be noted that while heralded as one of the successes of cognitive modeling [78], there are some areas of concern when considering applications of the power-law of learning. In work presented in [79], researchers have identified that the log-log linear model (see Equation 10) will not necessarily fit the power-law model of (9)

depending on the nature of the error in the collected data.

$$log(P(N)) = log(B) + \beta log(N + N_0) \tag{10}$$

As such, estimates of B and β can be improperly inferred from the log-log linear transformation of the power-law so while convenient, applying this transformation can introduce errors in the derived learning curves. The researchers suggested that non linear regression models must be used and the presented work follows this suggestion.

With this understanding of learning and learning curves in human study we now consider the situation where learning is applied to robotic students. In such a scenario, whether through observation or through interactive learning, the robot is learning the task directly from one or more human teachers. As such, we hypothesize that a robot learning curve can be extracted that follows similar characteristics to a human learning curve. We also hypothesize that this learning curve is related to the quality of the instruction provided. Finally, we believe that during the process of learning from the human interactively that the robot can extract enough information for it to generate a proxy of a learning curve.

2.3.2 Capturing the Learning Curve

For this work, teleoperation was accomplished by providing the operator with an overhead view of the robot. Based on visual cues, the operator demonstrated the target behavior and their actions were captured through the use of a joystick and passed on to the robot as they became available (near real-time). Data from the sonar sensors were used to represent the sensory state of the world. An off-board robot controller relayed all information between robot and operator, so the provided instruction is coupled with the appropriate visual cue.

The results presented in this section will be for a robot that is learning a wall following task in a maze environment. To measure the performance of the robot, a performance metric was devised. The metric, which was evaluated as a post-test

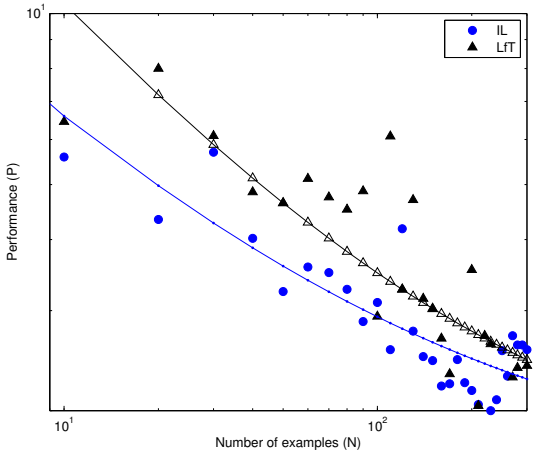


Figure 21: Learning curves for interactive learning and learning from teleoperation.

Table 2: Parameters for learning curves for learning from teleoperation, with and without interaction.

	IL	LfT
β	-0.55	-0.64
B	18.20	39.06
A	1.490	1.440

measure, incorporated distance (to evaluate task quality) and time of completion to generate the performance value. When teleoperating the robot (no learning), on average, expert human operators produced a performance value of 1.75 with the same metric. As the performance improves, this metric decreases in value.

The graph in Figure 21 presents curves that follow the characteristic shape of the "power-law of performance". As listed in Table 2, the magnitude of the learning rate observed for learning from teleoperation is larger than that of Interactive Learning but it is important to note that Interactive Learning out performs it for much of the learning process (see Figure 21). This surprising result occurs because more useful information is provided about the behavior earlier in the learning process. This means that with Interactive Learning, for the same number of examples, better quality instruction is presented earlier in the learning process. The larger magnitude of learning rate for learning from teleoperation and its larger learning range indicate

that its performance will eventually outperform Interactive Learning, as would be expected since the entire behavior is being presented. This expectation is confirmed by the smaller value of performance with "infinite instruction" which is captured in value A in Table 2.

This section confirms the hypothesis that the robot's learning curve fits the characteristics of human learning curves. While the asymptotic values are both smaller than average value for the human operator (1.75), these curves cross this value with more than 1500 examples and require an equivalent order of magnitude examples to improve performance by ten percent. The two remaining hypotheses will be addressed in Chapters 3 and 4, after a few more topics about the learning process are covered.

2.4 The Mechanisms of this Implementation of Interactive Learning

So far, the overview of the IL process has been presented in Section 2.1. The algorithms used have also been introduced in Section 2.2. In Section 2.3, the robotic learning curve was identified and the similarity between it and the learning curve for human subjects was recognized. What has not yet been done is to show how these components have been implemented and integrated into a cohesive strategy to make a robot move, or even learn. Also, while the reader has been shown how IL was successfully applied to learn some tasks, what has also not been presented were some of the qualities which make this type of learning difficult. In this section, both of these shortcomings will be addressed. This section will enable the discussion of the mechanics of IL to be concluded so that the usefulness of the process can be addressed. Subsequent sections of this thesis will present the research contributions of this work, but this section and those prior present useful information to position the reader to understand their meaning.

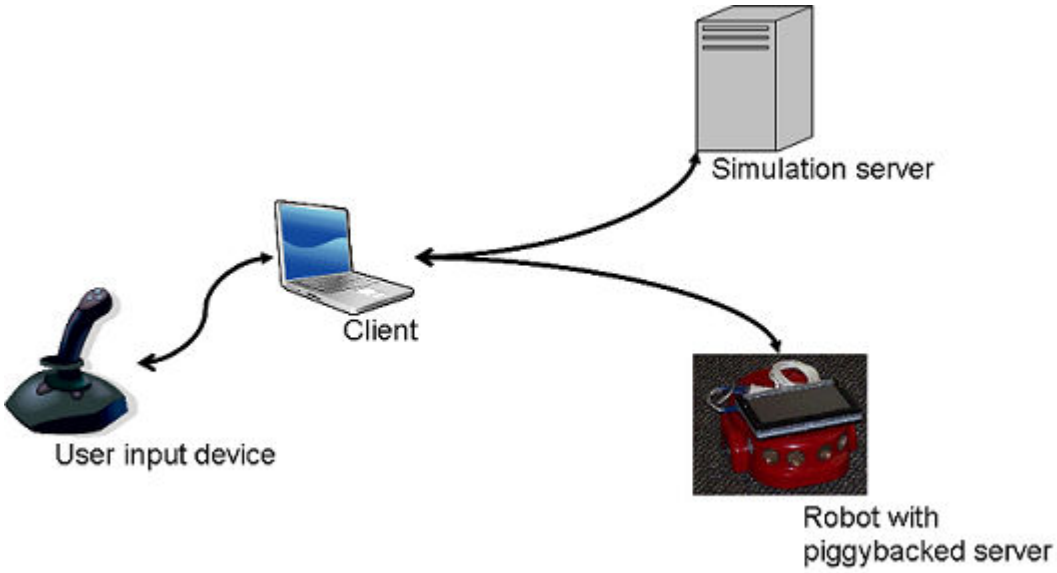


Figure 22: Control architecture. Picture indicates the major components and what modules that they communicate with.

2.4.1 Architecture

While MATLAB was initially a central component, the Player Project [31] was used as the primary mechanism of this architecture. Player was selected over other tools like Webots [102] and Microsoft Robotics Studio since it permitted easier interfacing with a wider range of devices. Player is also an open source project, thus it enabled study of the implementation. This permitted correction of, and additions to, relevant code in the corpus.

2.4.1.1 Gazebo

The simulator used for the bulk of this work is Gazebo. Gazebo is a 3D simulation environment which was developed as part of the Player project [31, 19]. Currently using third party libraries such as OGRE [68] and ODE [95], it permits multiple robots to be simulated and facilitates realistic interaction between objects in the virtual world.

Gazebo is becoming a cross platform simulator, but it is not at this time. All the implementations were realized with Ubuntu 8.04 or 9.04, a modern nVidia graphics

card, the nvidia-glx-177/180 driver. The base version of Gazebo which was used for the later stages of this work was revision 7322 of the code housed in the subversion repository [71]. This revision was patched with modifications developed within the Human-Automation Systems Lab. Most of the patches provided functionality required to mirror the form of the Pioneer3AT more fully in simulation. Patches were additionally developed to correct the implemented function of generic devices such as global position system, and the log reader/writer. Gazebo utilizes an xml configuration file at runtime which enables rich worlds to be generated dynamically. Environments are able to be generated from scratch, or to be imported from other formats through the use of applications such as Blender [11].

Gazebo is composed of several components or modules. The major components include the rendering module which is provided in major part by OGRE and the physics module provided in the main by ODE. There is also a controller module which handles the reading and writing to Gazebo interfaces. The sensor module, which uses data acquired from the physics and rendering modules, is used to provide simulated sensor data.

Additionally there is a player module which is used to interface the simulation environment with Player. There is also a shared memory interface to Gazebo called libgazebo. This interface permits a player server to access Gazebo. Finally, there is a server module which functions as the glue between the modules. This server module is different from the player server. This module provides the missing features/functionality needed for the player server to operate. The code base is not structured in the most logical fashion in all places, but it does lend itself easily to understanding how the major components function together.

This treatment is by no means intended to be an extensive explanation of this open source project and does not cover all the dependencies or prerequisites. What has been covered are the major components and how they function together.

2.4.1.2 Player Server

The language of much of the following description has been gleaned from various Player manuals as well as investigation into relevant source file for this open source project. The base version of player which was used for the later stages of this work was revision 7340 of the subversion repository. This revision was tagged as release-2-1-2.

The Player project provides a layer of abstraction which makes the process of interfacing with robots much more modular. The software is described as a "robot device interface" which seeks to provide a robotic operating system. Just as computers have benefited from abstracting away the relationship between devices like mice and touch-screen from applications like word processors and web-browsers, Player provides the mechanisms that permits code to be written for generic interfaces which can be provided by specific devices as needed. In this body of research, robots utilized the P2OS/ARCOS family of embedded controllers but the instruction provided could be used with other robots for which the necessary interfaces exist.

In most cases the interfaces of interest are the position2d interface, the camera interface, and the actuator array interface. Position2D interfaces allows ground-based robots to accept commands which control the motion in terms of translation and rotation. This interface also provides access to estimates of position based on wheel encoders. These estimates of notoriously sensitive to drift and without intelligent techniques like Kalman filtering, are usually of little value.

As can be expected, the camera interface permits images to be acquired from attached camera devices. The actuator array interface is another relevant interface provided by the Player project. This interface enables each of the motors and the 5 DOF arm to be controlled. This interface can also be used with other devices which are in essence collections of motors.

Player is composed of two portions, one responsible for the transport layer, and the other for the core functionality. This core functionality is also provided by two

complimentary sections of code, the core library - libplayercore, and the built-in driver library - libplayerdriver. The transport layer currently only implements TCP [] and is thus comprised of libplayertcp and libplayerxdr. libplayertcp establishes socket connections and manages transmission of messages across these sockets, libplayerxdr is responsible for platform independent encoding of the messages which are being transmitted. There is also two other libraries that should be mentioned, libplayererror and libplayerutils. These libraries provide useful housekeeping functions which aid the coding process. More information about Player can be found in [19]

2.4.1.3 Client

Modularity was a design principle adhered to as much as possible throughout the implementation. The components of the client of comprised of the brain, the brain stem, and the player client. The player client is accessed through the library from the player stage Gazebo project which is compiled with the brainstem. The remaining three components are based on custom written code.

The brainstem serves as the glue for the software ensemble. It is written in C++ and links the human teacher through the Human Interface Device (HID), and the robot (or the simulated robot) through the player client. As mentioned earlier, the brainstem is linked to the player client at compile time so the single executable exists for the two bodies of code. Through sockets, the brainstem is also connected to the brain. The sockets provide a channel through which queries issued and responses are received related to the current behavior being executed. The same channel is utilized to train the brain however the data is encoded in a slightly different manner.

The data sent from brainstem to brain is an aggregate of all the current sensor and actuator data. In the case of the vision modality, the values transmitted are not raw values but the result of a series of transforms which will be detailed in Section 2.4.2. The primary functions of the brainstem are to convert sensor values and actuator

values between the formats required for processing in the brain, formats used by the robot, and formats used by the HID.

The C++ brain:- The purpose of the brain is to process and capture and store knowledge acquired through interaction; and to produce responses to queries issued by the brainstem. In the C++ brain, to enable high throughput, threads are used to implement the brain. Using the Pthreads [63] library, a base thread is created to receive and respond to the brainstem. A child thread is spawned to update the current state of the brain as needed. The new instruction incorporated as soon as the update that can accommodate it. Depending on the timing of new arrivals, it is possible that two or more items may be incorporated into the same batch. It is more often the case however that only a single data item is incorporated into the knowledge base at the time. The SOM class implemented is a modified form of the work published in [99].

The Octave/Matlab brain:- The core of the Matlab/octave brain is the SOM toolbox provided in [45]. While this version of the brain is not threaded and thus cannot capitalize on the advantages of parallelism, the implemented functions are in matrix form thus take advantage of the speed benefits provided by these numerical packages. This version of the brain is only 8 to 10 times slower than its C++ counterpart.

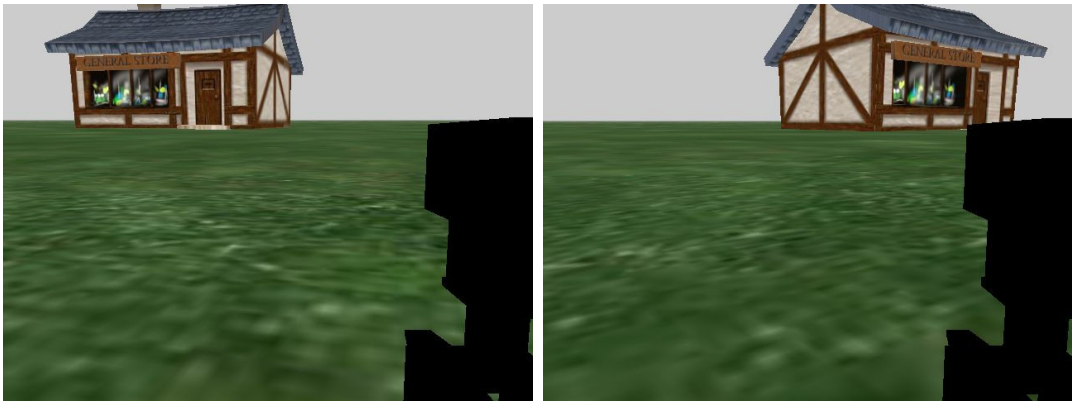
2.4.1.4 Human Interface Devices (HID)

To gather information and control signals from the instructor, the primary HID used was a game controller. This device, recognized as a joystick in most modern operating systems, was incorporated either through the Java joystick driver or via the joystick interface provided in Player.

2.4.2 Treatment of Vision as a Modality

Influenced by the work of Goldberg [104], our incorporation of vision as a sensing modality uses a fast first stage where images are reduced to a signature. This signature has the property that two similar images have signatures which are more alike than the signatures generated for dissimilar images. Once the signature is generated, it is treated like any other sensor input and a SOM is utilized to transform the sensor data into a sensor state. Since the SOM is topologically preserving, the sensor states for two similar images are also closer together than the states for two different images.

While similarity between signatures and thus the similarity between images is of interest, it should be noted that unlike many approaches, object recognition is really not the goal. For example, consider the two images in Figure 23. It is less important that the object is recognized in both images than the correct action is taken when either view is presented. It may be the case that the robot should learn to approach the door slowly when the image shown in Figure 23(a) is observed, If it is in a different location relative to the the building (say that presented in Figure 23(b)), then it should have learned to find the door that is painted green (not shown in image).



(a) View 1. (b) View 2.

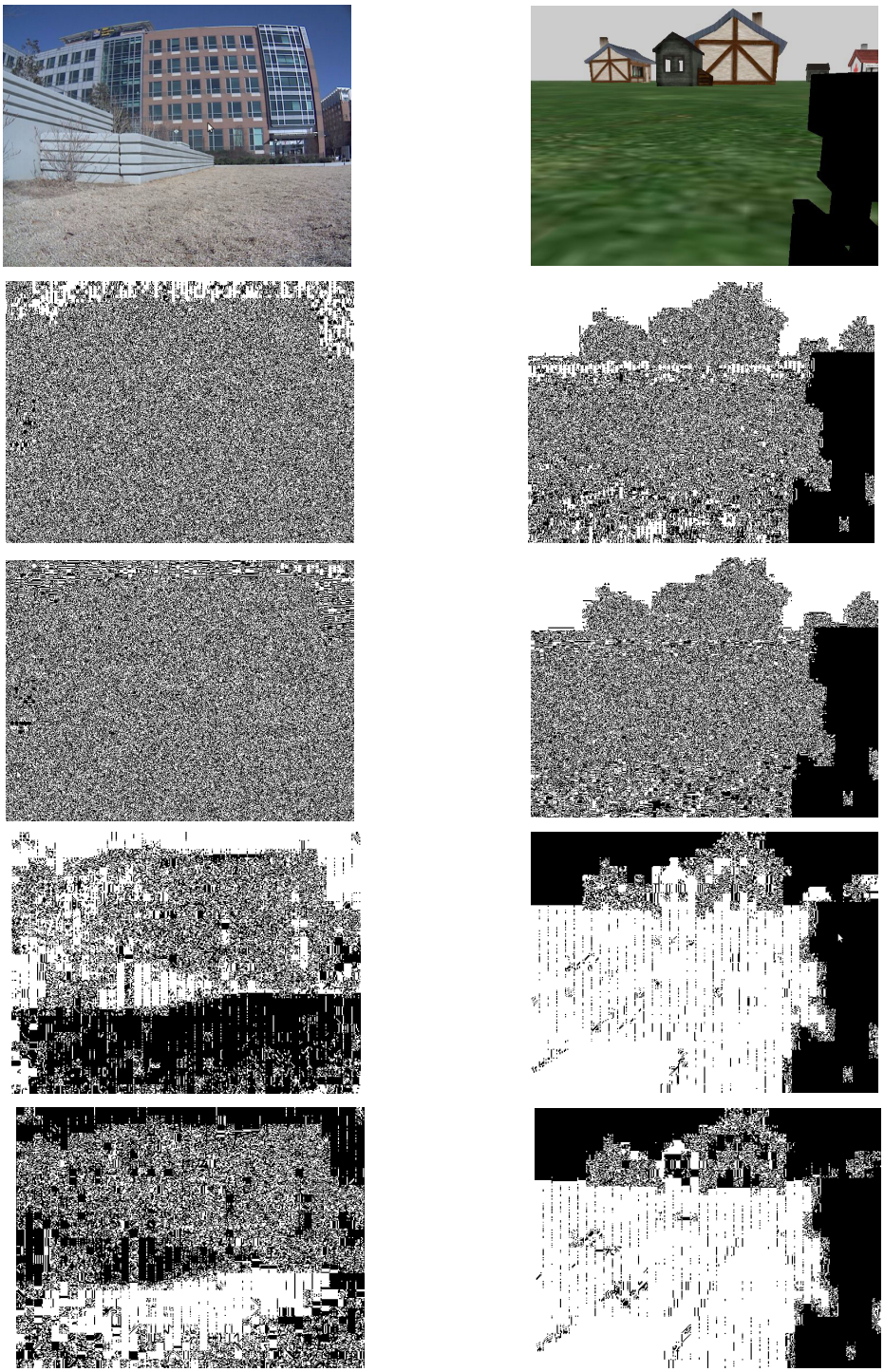
Figure 23: Two views of the same building in the simulated outdoor environment provided by Gazebo.

The underlying concept which is being leveraged is that for an embodied agent, things look different if the relationships between it and the other items in its environment are different. If the difference is observed to be irrelevant, then the actions taken will be equivalent. Such knowledge is acquired solely from the provided instruction. In this case, the robot's aim is not to reduce the set or to extrapolate, but to quickly recall and apply the recalled information.

This means that rotation and translation invariance are not useful for this application. Other systems or subsystems may be able to apply such properties and these can complement this approach but directly they serve only to artificially reduce the relevant spaces which should be learned.

Inspired by the work presented by Ng [57, 84], where it was leveraged to facilitate monocular depth perception, a texture-based signature generated primarily from the application of the Laws masks are utilized (See Appendix A). The signature uses energy uncovered by different types of filters to develop a measure of image content. In figure 24, the result of applying four of the filters are presented when applied to the image in figure 23(a) and another image taken from the real robot while outdoors. The response of the filters differ greatly, so when these images are aggregated by summing the responses in each of the 16 regions, a different set of 16 numbers is generated. Through this process, each image is converted to an image signature. Seventeen convolutions are applied and generate a series of 272 values which are then converted into a sensor state as previously outlined.

In the previously discussed experiments, and those discussed in the remainder of this thesis, this architecture functions as the primary mechanism for testing and evaluation of the IL methodology.



(a) Real. (b) Simulated.

Figure 24: Two sets of filter responses from two different picture sources, one from data from a real robot, the other from a simulated robot in a three dimensional world.

CHAPTER III

EXTRACTING INFORMATION FROM INTERACTION

Now that the process of interactive learning has been presented in some depth, attention is now focused on the student and how they can extract information from this process. Again,this is important since the purpose is to equip the robotic student so that it can reduce the operational requirements for their teachers/operators. It was proposed that a robotic student can extract information about the interaction it receives, and this information can provide relevant insight about the learning process. It is this insight which can be used to aid the learning process. It was also proposed that this insight could be generated without adding additional responsibility on the human teacher. The acquired insight should serve to aid the human teacher by equipping the student to access receive instruction, enabling the student to generate useful feedback, and permitting the teacher greater freedom during the interaction (teaching process).

For interactive learning to be effective, it was found that the teaching process must have consistency in the instruction. This property, which was identified as critical to the process of interactive learning is called coherence. Coherence is a term which relates the consistency between two types of related information. These sets can be both current, or one or both of them could have been previously acquired. Coherence was observed to have three forms. The first is between sensor and action data. The second is between subsequent snapshots of the behavior as it is learned. And finally, the third is between examples of different behaviors. Each of these classes of coherence will be presented in greater detail in the following three sections as experiments exploring these claims are considered.

3.1 Extracting Information from Interaction - Classification Error

In order to understand coherence, we must first define parameters which can be used to quantify the property. The first considered parameter is denoted as the change in classification error as instruction is provided. As described in Chapter 2, adaptive classifiers are utilized to map the input data to its associated input state. Since this is an adaptive process, observing how the error changes over the course of interaction can provide insight on how well the classifiers are adapting to the provided data. Succinctly put, if the error is not increasing, this implies that the behavior is being learned effectively. There is more to this process than this simple relationship, but it captures the crux of the matter. To explore this relationship in more detail the following experiments are presented.

3.1.1 How to Access Error Data?

In the experiments presented in Section 2.2.4, Interactive Learning was applied to teach the robot to perform the desired tasks. As the teacher provided examples in each case, the observed error can be measured. The error that is represented by the difference between the weights of the best matching unit (BMU) and the current input is referred to as the quantization error.

The mean quantization error (MQE) is the mean value of the error for all the input values and their associated BMUs. This MQE is thus a measure of how closely the map is able to classify the provided input values. Pictorially this is represented by the length of the arrows in Figure 25. The equation presented in (11) defines this term. In this equation, u is the number of classes, n is the dimension of the data, and m_i is the BMU in the classifier for the data item a_k. The value of a_k is either x_k or y_k depending on if the MQE is being calculated for sensing or actuation data.

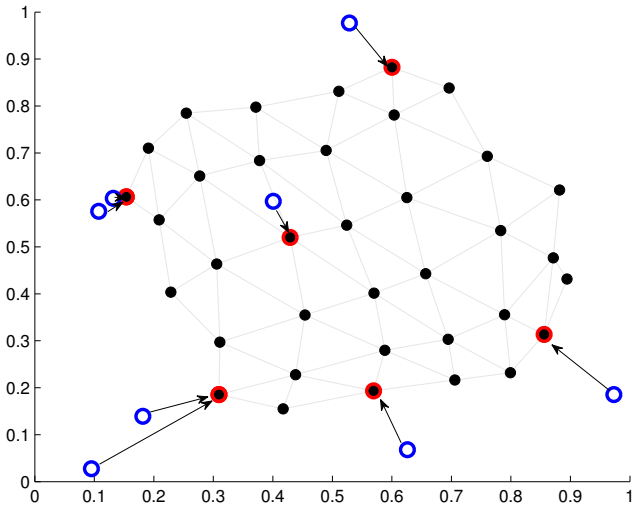
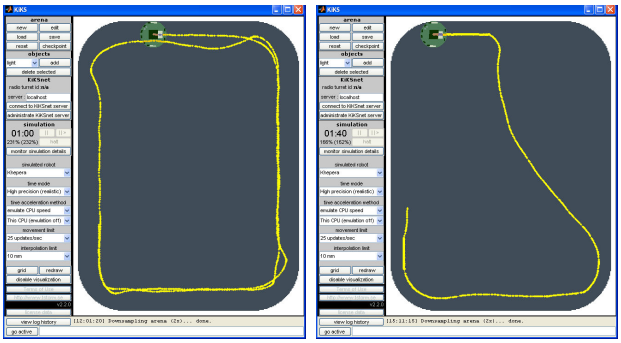


Figure 25: The SOM trained on a sample dataset. The blue circles indicate new data, and the red circled nodes, the BMUs of the trained map for this new data. The error in classifying the new data is represented in the distance between the data and their associated BMUs.

$$MQE = \frac{1}{u}\sum_{k}\frac{1}{n}||m_i - a_k|| \tag{11}$$

3.1.2 How Does this Error Look?

To gain a better sense of what value MQE can provide, this quantity is considered for two scenarios where IL was applied to learn a task. These scenarios, which both utilize instruction from an expert user, are presented since they both capture cases where learning did indeed occur. More importantly, they are also presented since they provide a look at two cases, one where "good" instruction was used in the learning process (Scenario A), and another where the provided instruction was perturbed in a controlled manner prior to being incorporated into the learning process (Scenario B). sIn this latter case, the instructor was required to provide more examples of the task under consideration because, as expected, the student (i.e. the robot) was not able to learn the task as easily as in the former case.



(a) Scenario A. (b) Scenario B.

Figure 26: Robots demonstrating learned behaviors for the two described scenarios.

The path demonstrated by the student after each of these scenarios concluded is presented in Figure 26. This figure shows that in both scenarios, the robot appears to be demonstrating the desired task, in this case wall following. The path in Figure 26(b) shows that in that scenario the student did suffer some ill effects of learning from noisy data.

As mentioned, the point of this exercise is not to show that the student still learned in spite of the challenges, but to observe how the MQE changes in this scenario. The MQE plots shown in Figure 27 capture both error and interaction information. Unlike the error information, which is explicitly plotted on the y-axis, the interaction information is captured in the density of data items. The presence of each data item indicates that instruction was provided at that time step.

For both scenarios, the process of interactive learning began after 500 time steps. Then, starting from zero initial knowledge, the MQE increased significantly until the maps were able to capture most of the provided data. After this point the MQE enters a phase where it increases at a slower rate, as the maps are fine tuned to classify the data.

Each of the MQE curves follow this basic trend. Where they differ however is the level of the MQE for which the phase change occurs. For scenario B (red circles), the MQE at the phase change is over two times that for scenario A (blue points). Another

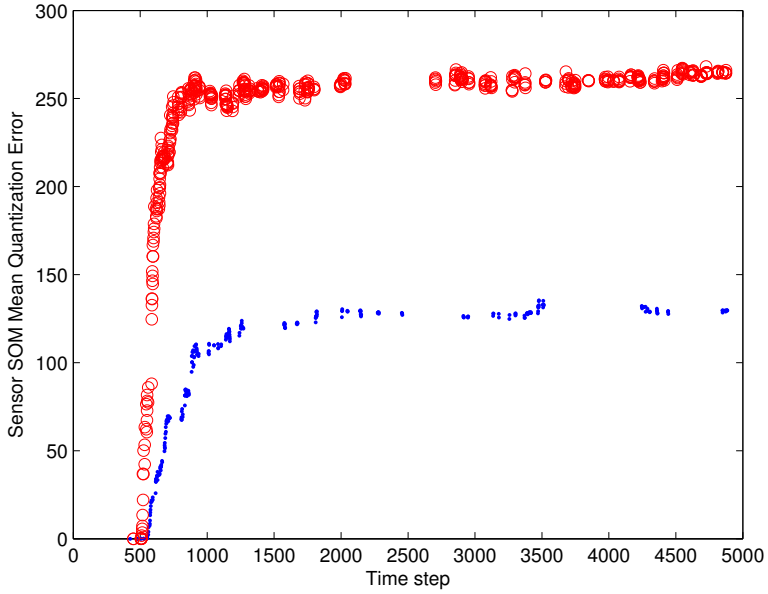


Figure 27: The mean quantization error graph for the sensor SOM in two scenarios. In Scenario A (blue points), the learning process occurred with an expert user. In Scenario B (red circles), the same user provided instruction but the instruction was augmented with an additive noise.

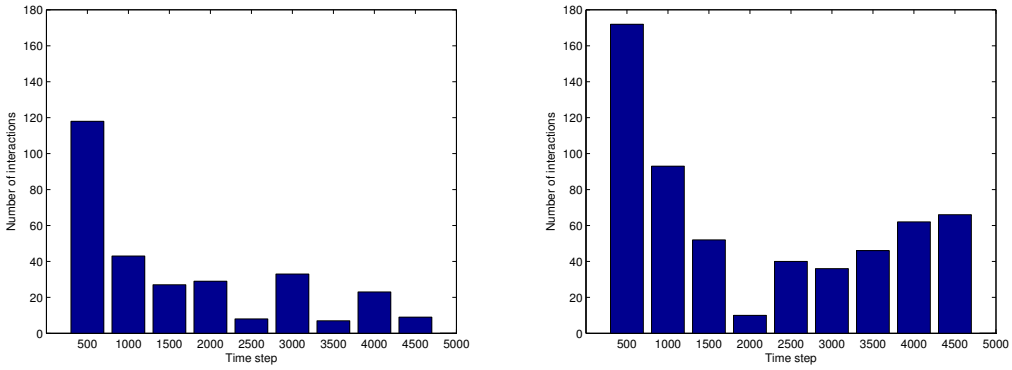
area of divergence is during the latter phase of scenario B, the MQE increases more than it does in scenario A. This is reflected by the larger slope observed in this phase.

3.1.3 What Information Can These Properties Provide?

The point density of the graph when projected onto the x-axis indicates the interaction level. The slope of the graph indicates how the error changes as additional examples are presented. As made more readily evident in Figure 28, the point density in the MQE plots both decrease over the course of interaction, one more so than the other. This reduction indicates that the human instructor observing the student's progress was less inclined to provide instruction as time progressed. Assuming that the instructor remained faithful to the task of teaching the student to perform the task, this change would imply that the student's performance was improving.

While we assume that the instructor's goal is to teach the student a specific task, we do not assume that the instructor is actually a good teacher. It is however a

claim that this research can identify properties of teaching based on the provided instruction. These properties are linked to the amount of interaction that occurs as evidenced by histograms like those shown in Figure 28, as well as how the MQE changes with additional instruction.



(a) Scenario A.

(b) Scenario B.

Figure 28: Histograms of data for each scenario. The bin size is 500 timesteps.

Before discussion of how changes in MQE can be evaluated, it must be recalled that the purpose of instruction is to indicate to the student the intended action for the current state. The SOMs adapt their form to best represent the instruction provided. If the sensor (or the actuator) data presented is not well classified by the map, it will update to better capture this new data item. Once the map is updated, the MQE is then calculated to see how well this more recent update enables the map to effectively classify the provided instruction. As could be expected, the MQE is calculated for each SOM. If the MQE increases continually over some small window, this indicates that additional instruction did not help to map to better classify the instruction. If the MQE continues to increase significantly, this would indicate that the considered data is not at all well represented by the map.

The magnitude of the MQE also contains valuable information. The magnitude provides an indication of how well the map captures the instruction. Unlike studying how additional instruction changes the MQE, utilizing the magnitude of the MQE

requires additional knowledge about the relationship between the value of the error and performance.

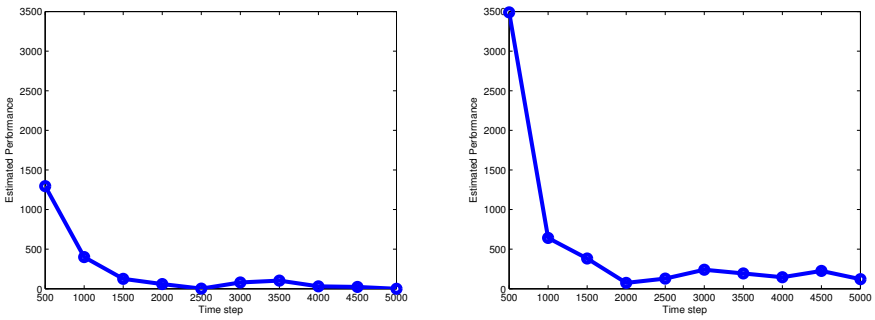
For this work, additional information such as this is unavailable to the student, and is likely also unavailable to the instructor in a quantifiable form. Studying changes in MQE is only possible by making assumptions about the process of interaction, and fortunately, such assumptions are verifiable based on other properties of the interaction (density). To move this discourse from the qualitative to the quantitative, the equation in (12) is presented. In this equation, a predictor of performance in a given window is generated by observing how many times the MQE increases over that window.

$$\text{prediction}_j = \sum_{p=Beg_j}^{End_j-1} max\{MQE_{p+1} - MQE_p, 0\} \tag{12}$$

In this equation, Beg_j and $End_j - 1$ mark the first and the penultimate examples during the j^{th} window. MQE_p is the error calculated after the p^{th} example has been provided. The contribution from the sensing and action SOMs are combined by assessing the norm of both predictions.

Reductions in MQE are not considered as part of this metric since the purpose of additional instruction is to correct improper past action. This metric would have to be modified if negative instruction (examples of what not to do), or feedback about correct performance were incorporated into the interaction process. Such changes however are outside the scope of this research.

Based on (12), the plots shown in Figure 29 represent the estimated performance for learning in scenarios A and B. This predictive metric is presented as information is extracted from interaction. This metric was derived from the classification error and permits information to be extracted without details of the task being taught, knowledge about the teacher providing the instruction or even insight about the sensors or actuators with which the student was endowed.



(a) Scenario A. (b) Scenario B.

Figure 29: Predicted Performance vs. # laps for interactive learning under two circumstances.

3.1.4 Changes in the Pattern of Mean Quantization Error Types for Different Users

In the previous section, the differences in the ways that the MQE data change over the course of interaction are inferred to be functions of the instruction provided. The two scenarios considered presented different circumstances, one where "good" instruction was used for learning and in the other "bad" instruction was used. It was shown that the student could be equipped to estimate useful information about the nature of their performance. This performance is affected by the quality of the provided instruction. Quality instruction is not just determined by the conditions under which learning occurs. It is also affected by the ability of the teacher providing the instruction.

To study how the information can vary with different types of teachers, a placeholder simulation is considered. The placeholder simulation is a reduced form of the Oz of Wizard [91] study. The information generated from classification error can be effectively studied using this approach as will be demonstrated shortly.

Due to its flexible structure and ability to map non-linear functions, a neurocontroller is used to provide expert action. The neurocontroller, an ANN based technology was tuned to perform the target task (π_i). The inputs are the robot's

Table 3: Parameters presented in [26] to characterize stereotypes of three types of robot users. These parameters are now used to generate simulated instruction from these types of users.

Type	% Accuracy (α)	% Expertize(β)	Processing delay (τ)
Novice	30	30	15
Scientist	50	100	15
Expert	100	100	0

sensor values and its outputs enable actuation. Since the object is to learn from human demonstrations, this actuation is then used by the robotic student to gain more information about π_i and adapt its approximation of the behavior, $\hat{\pi}_i$.

Stereotypes of human actors (novice, scientist, and expert), leveraged from research presented in [26] are used to represent the sources of instruction. When the parameters which define these stereotypes which are reminiscent of the roles outlined by Sholtz [87], were initially presented, they were not applied in a placeholder simulation. They were applied to cases where automated agents performed decisions based on support from members of each class of stereotype. This is not a limitation however since the same intuition used to assess the decisions made, is applied to determine how a member from a class would act.

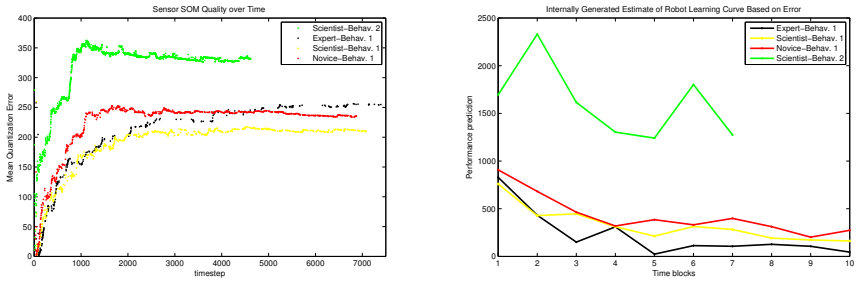
In the model presented, each stereotype is defined by three parameters: the accuracy in action, the expertize in determining the correct action, and the time delay (or processing time) needed to determine the required action. In Table 3 numbers which characterize each stereotype are presented.

To convert these stereotypes into a useful form for learning from demonstration, actuation is considered to be normally distributed with mean m and variance σ. The mean m is defined to be a function of the user's expertise (β) while σ a function of their accuracy (α). The mean m is further defined as a random variable $N(a, 1-\beta)$, where a is the actuation provided by the expert user. This applied method is summarized as actuation having the following distribution: $N(N(a, 1-\beta), 1-\alpha)$.

Finally to model the teacher's propensity to provide instruction the "action potential" is modeled. The larger the difference between the expected behavior and the observed behavior, the more likely it is that the teacher will notice the discrepancy. This would mean that it would be more likely that the teacher would provide instruction to the robot. If the demonstrated behavior is very close to what the teacher would have been likely to do, the it would be quite unlikely (but not improbable) that the teacher would still provide instruction to the robot. To model the essence of this relationship between action and perceived difference between observed and expected action we apply the ERF function (as defined in Equation 13). For this work the teacher was assumed to be slightly risk averse so the probability of action was increased by scaling the error by $\frac{1}{50}$. Risk aversion is a property which indicates that the teacher's action is influenced by perceived risk and they are strongly motivated to avoid such risk. By acting more frequently, the potential to encounter harm is reduced.

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_{t=0}^{z} e^{-t^2} dt \tag{13}$$

Figure 30(a) shows the mean quantization errors measured for the sensor classifier as instruction was provided. In all cases the graphs show that the classifiers were eventually able to classify the data since the MQE did not keep increasing as more examples were provided. Prior work has shown that these graphs can be used to identify conditions such as "over training" and "contradiction in provided instruction" (which could be considered malicious instruction). The focus is limited however to feedback which indicates the degree of success in learning. Finally, it is again assumed that the teacher is providing the best instruction possible when needed. The same infrastructure used to acquire instruction from human teachers is connected to this simulated human instruction "engine" and instruction is provided to the student.



(a) Mean quantization error. (b) Prediction of performance.

Figure 30: The data gathered and prediction generated for each of the stereotypes of human performing the following task (yellow, black, and red curves). The data in green was generated for a different behavior (avoid). This figure indicates that can be variations based on the type of instructor, as well as based on the target task.

Figure 30(b) also shows the learning curves generated for each considered stereotype. While comparison of absolute values between cases can be used to discuss the advantages of learning from experts over learning from novices, what is more important is to note that in each case it is possible to identify that some degree of success occurred. The difference between initial estimate and the estimate at any time is a measure of how much learning has occurred. Due to the approach used, the best possible performance estimate is zero so it is also possible to identify how much improvement is possible. It is important to again note that here we assume the teacher is doing their best to provide good instruction and to correct the student whenever necessary.

3.2 Extracting Information from Interaction - (Empirical) Entropy

Unlike classification error, which is an intrinsic property of a component used to facilitate IL, entropy is a derived property linked to the manner in which the behavior is represented and implemented. To present how information is extracted from interaction using entropy (IfI-EE), first a brief recap is presented for behaviors. Next the property (entropy) is defined first qualitatively then quantitatively. And finally, the process of extracting information from the data is presented.

3.2.1 Behaviors

As introduced in Section 2.1.1, a behavior is represented as a two-dimensional mapping which links sensor data to the associated action data. SOMs are used to reduce the dimensionality of the sensor and the action data to the two dimensions but the behavior still remains a discrete mapping between these domains.

As discussed, for any sensor state there is a distribution of action states which are associated. This distribution is based directly on the instructions provided by the teacher. If the teacher never demonstrates action a_j when the sensor state s_q was presented, then a_j would have zero probability of being selected by the student. If however a_l, a_m, and a_n were each demonstrated when that state was presented, the distribution would link the probability of any of their selection to the frequency of occurrence during demonstration.

For this work it is assumed that each behavior has an underlying and unknown deterministic mapping between sensing and action. The goal of the learning process is to uncover this mapping based on examples of the behavior. It is also assumed that an example of the behavior is at best an approximation of the actual behavior. Since instruction is provided interactively, an example is provided incrementally and thus it is a partial approximation which becomes more complete over time. When taken together, all of these statements indicate that each example can be interpreted as a statistical sample from the distribution of mappings which are approximations of the intended behavior. It is this 2-D distribution of which entropy is calculated.

Figure 31 presents the mapping generated when a teacher taught the student to perform a "wall following" behavior. There are 100 possible sensor and actuator states which respectively classify data that is eight and two channels of data. For the sensor data, each channel has a range of 0 to 1024 with a resolution of 1. Each channel of actuator data has a range of 10 to -10 with a resolution of 0.01. The sensor data is thus aggregated into a term which can have 2^{13} possible values while the actuator

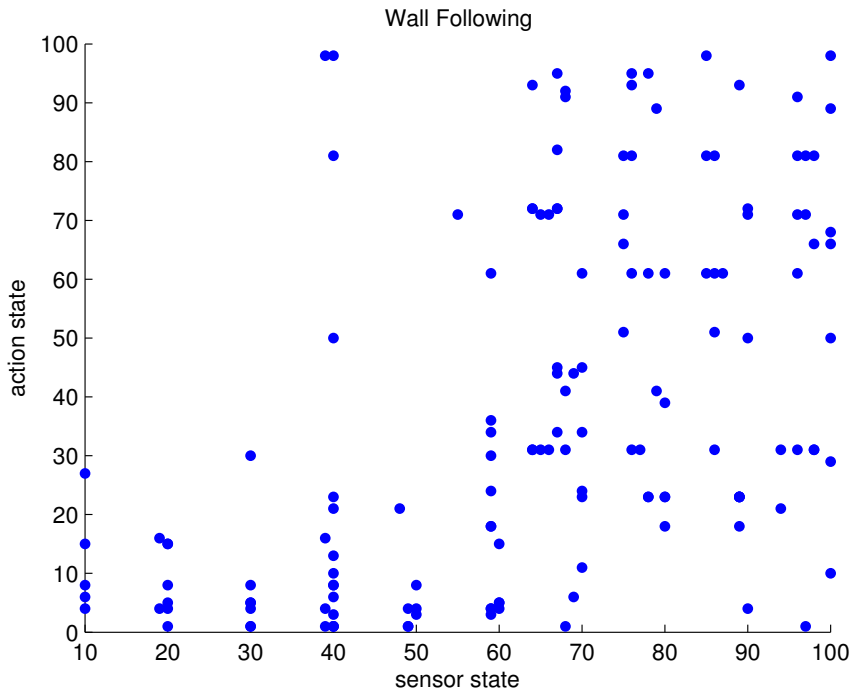


Figure 31: The base representation of a behavior, the mapping of sensor states to actuator states. In this behavior which was a navigation task, 100 states were used to map both sensor and actuator data.

data is aggregated $4 * 10^3$. These aggregate quantities are mapped into the sensor and actuator states. As is shown in Figure 31, for this behavior, while they are equally possible, all combinations of sensor and action pairings are not equally likely.

3.2.2 Entropy

The concept of entropy (more formally information entropy) has its roots in information theory and has been used to derive many important results. There is much that can be presented on the topic, but a more thorough development can be found in Chapter 2 of [34]. In this research, entropy based on "finite alphabet random processes" is considered exclusively since the number of sensor and action states is finite.

Entropy is a property which captures the information content of a random variable. Often referred to in the form of the average number of bits needed to encode

information, entropy captures a measure of disorder. In terms of its application in this work, entropy provides a measure of how specific a mapping a behavior appears to be. For example, a given sensor state, s_q, for which a large number of action states have been demonstrated does not demonstrate a very specific relationship between sensing and action. Given that s_q was observed, there would be a relatively small but non-zero probability that any of these action states will be selected. If however, there were very few actions demonstrated and these actions were demonstrated frequently when s_q was observed then there would be less uncertainty about the action that would be selected.

In this research, entropy is harnessed through the capture of the empirical distribution of the behavior. It is recognized that the empirical distribution isn't always the best estimate of the true distribution, but the empirical distribution is readily available and can be acquired and updated quickly without a great deal of background information. This last property, the flexibility in the light of a dearth of information, is very crucial to this research area since for the target benefactors, freedom from a priori knowledge is a boon. Entropy calculated in this manner is often referred to as empirical entropy.

The equation used to calculate the entropy of a behavior is presented in (14). In this equation, $p(i)$ represents the probability that i is present in the provided instruction set I. The quantity i represents an action-sensor state pair like (x_k, y_k) presented previously in Equation 3.

$$H = -\sum_{i \in I} p(i) ln\left(p(i)\right) \tag{14}$$

In this equation, the convention that $0 * log(0) = 0$ is assumed since as $\lim x \rightarrow 0$, $\lim x * log(x) \rightarrow 0$. Intuitively, this is also acceptable as one would expect that an event of zero probability to contribute nothing towards entropy. Since the action state selected is based on the sensor state observed and it is assumed that the instructor

intends to teach a specific behavior, sensor and action states are not statistically independent quantities. While it it true that the actual action state(s) associated with a given sensor state can be arbitrary (it is up to the instructor), independence does not hold. As such, the maximum entropy is bounded above by the sum of the maximum entropy contributed by both the sensor data and the action data, $H_{max,s}$ and $H_{max,a}$ respectively. This relationship is captured in Equation 15.

$$H_{max} < H_{max,s} + H_{max,a} \tag{15}$$

Since the number of sensor and action states are fixed (and known to the system), the maximum entropy contributed by sensing and action are both known. This permits the upper bound to be calculated and it is this term which is used to normalize the entropy values calculated. In Figure 32, graphs of the entropy calculated under the different human stereotypical teachers. As in the previous discussion of extracting information from classification error, a reduced Oz of Wizard [91] approach was applied to simulate instruction. The data presented was also captured for three classes of instructors teaching one behavior (following) and one instructor, the scientist, teaching a second behavior (avoiding).

3.2.3 What Information Can This Property Provide?

Again it is important to consider the value of studying entropy and what information it can provide. As with the MQE, it is expected that this property will initially increase as examples are presented but as the student is better able to demonstrate the task, the instructions provided will have less impact on the overall process. Subsequent instruction fine tunes the student‘s behavior, but if learning is successful, radical changes are less likely to occur as time progresses. This trend is observed in Figure 32 and the interpretation is captured in the learning curves presented in Figure 33. These learning curves corroborate the information from the curves presented

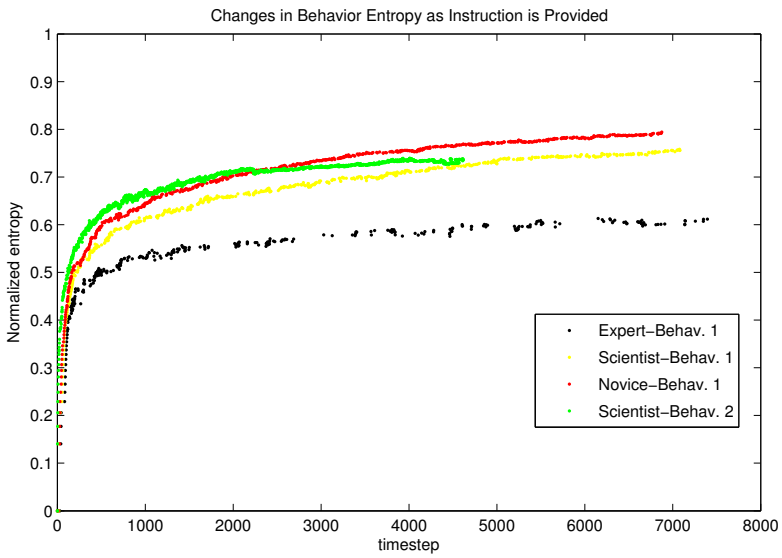


Figure 32: Empirical entropy calculated for the instruction provided by simulated human actors in various conditions.

earlier in Figure 30(b). While this treatment utilizes the change of entropy over time which some may refer to as the entropy rate, that term has already been applied. *Entropy rate* is used to refer to the mean entropy of a random variable [34].

Revisiting Figure 32, the value of the normalized entropy also contains information which can be used for feedback. The steady state entropy value captures the degree of disorder present in the provided instruction. The larger this value, the closer the behavior is to pure random action. The robotic student‘s ability to quantify this quality could thus be used to express its degree of confidence in its teacher‘s instruction.

3.3 Extracting Information from Interaction - Statistical Similarity

Learning systems have been successfully applied to many aspects of robotic systems. One example of such includes [38] which demonstrated a robotic arm that was trained using a neural network and examples of picking up a single object. Another example [83], demonstrated a more general algorithm to enable a robot to identify the grasping

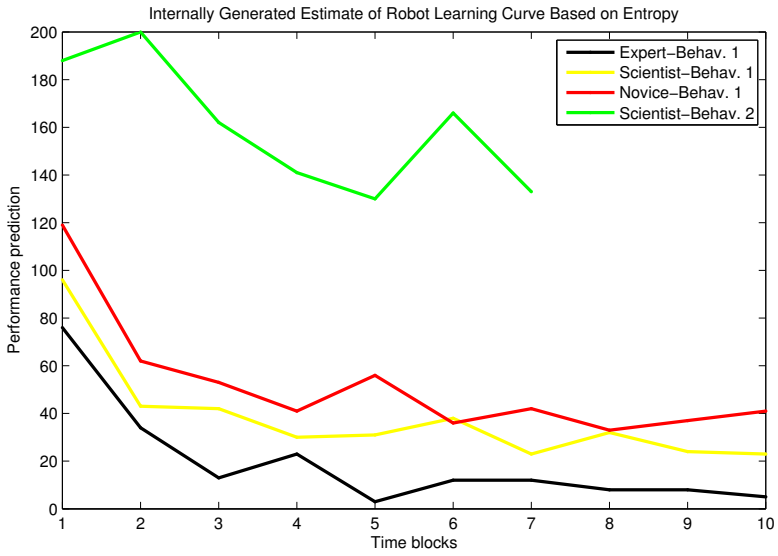


Figure 33: Prediction of performance based on the entropy data collected.

point at which an object should be lifted.

While these two examples arise from robotic manipulation, they represent a larger class of learning application, those which rely on a training set to provide information about the task under the study. Few would argue that improving the quality of the training set would have desirable outcomes for the learning process. Some of the ideal outcomes include reducing learning time, improved performance, reduced memory overhead, and more direct learning progress. If elements of the training set that did not positively impact the outcome of the learning process could be removed, it is likely that such desirable outcomes can be achieved. This is one method to improve the quality of the learning process.

We believe that coherence is the property that would enable the filtering of behavior sets (training sets). As presented previously, coherence can be used to identify relationships within sensor data or within actuator data.And in this section, the third approach taken to quantify this property is presented. This parameter, denoted as statical significance, can be quantified though the use of goodness of fit tests which are presented next.

3.3.1 How to Acquire the Required Data

Goodness of fit tests are statistical hypothesis test which are used to determine whether a set of observations $x_1, x_2, ..., x_n$ are an independent sample from a particular distribution F. The specific hypothesis, the NULL hypothesis H_0, is stated as follows:

$$H_0 : \text{The } x_i\text{'s are IID random variables with distribution } F \tag{16}$$

The result of these tests is to determine whether the hypothesis is rejected or not. Failure to reject, while tempting, is not the same as acceptance. So if the test rejects the hypothesis, this indicates that the x_i's are **not** IID random variables with distribution F. If the test fails to reject the hypothesis this does not mean that the x_i's are IID random variables with distribution F, but there is not enough evidence to indicate that they are not.

When applying this construct to consider behaviors, the x_is represent the set of demonstrations of behaviors. As previously shown, each example of a behavior is considered to be a sample from a distribution. This distribution contains approximations of the unknown, underlying, deterministic mapping between sensing and action. In this case, the hypothesis tests whether the observations are independent samples from the same behavior distribution.

3.3.2 Types of Tests

There are three major goodness-of-fit tests: the Pearson χ^2 test [65], the Kolmogorov-Smirnov test and the Anderson-Darling test [65]. With distributions such as those considered in this work, the Kolmogorov-Smirnov test is the most readily applicable [73].

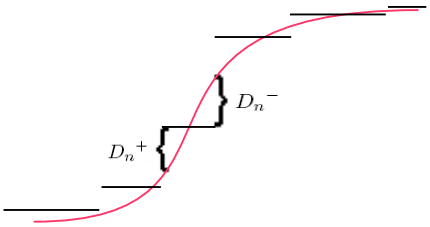


Figure 34: Distribution plus the distribution function

3.3.2.1 The Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) test is a widely used "goodness of fit" test. It can be applied in two ways, the one sample and the two sample fashion. The one sample K-S test investigates the likelihood that a given sample was generated from a specific model. The two sample K-S test identifies whether two samples were generated by the same distribution.

It is useful to note that this test compares the distributions of data and not the actual data itself. In the one sample version of the test, the empirical distribution function generated for the provided data is compared to the distribution function of the hypothesized distribution. In the two sample format, the two empirical distribution functions are considered.

This test however, does not require that the data be grouped into bins so there is no lost information, there are no intervals to define. It is valid for any sample size n, and while more limited in applicability, this test is also more powerful than the χ^2 test [47].

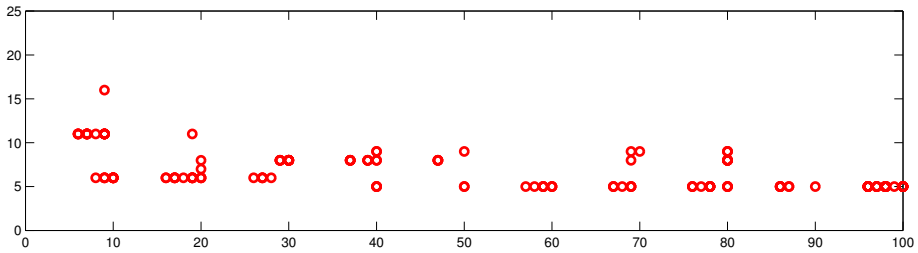
$$D = \sup_x\{|F_n(x) - \hat{F}(x)|\} \tag{17}$$

The statistic D represents the largest distance between $F_n(x)$ and $\hat{F}(x)$. To more easily understand the definition of D, two additional quantities ${D_n}^+$, ${D_n}^-$ are defined. ${D_n}^+$ is the maximum distance that $\hat{F}(x)$ is above $F_n(x)$, while ${D_n}^-$ is the maximum distance that $\hat{F}(x)$ is below $F_n(x)$, both of these evaluated over the range of X.

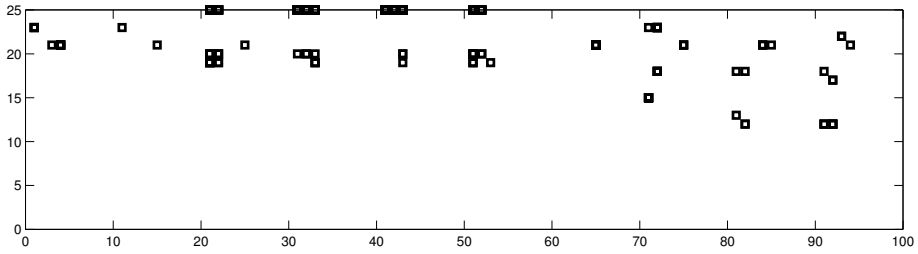
$$D = \max\{D_n{}^+, D_n{}^-\} \tag{18}$$

Figure 34 presents a graphical representation of this definition for a sample case. It is this statistic which is used to evaluate the hypothesis. If D is large, the comparison between the two distributions is unfavorable, while if D is small the distributions are more closely aligned. How large is large? How small is small? These answers depend on the distribution and there are tables generated from Monte Carlo simulations which indicate the thresholds and the conditions under which they are applicable [58].

3.3.2.2 The Two Dimension Kolmogorov-Smirnov Test



(a) Follow.



(b) Avoid.

Figure 35: Two behaviors (distributions) which will be used to explain this test.

The pictured representation in Figure 35 helps to explain the process used in applying the statistical test. If these two behaviors were generated from different distribution functions, they would not share similar statistical properties. In one dimension the pdf of random variable captures all the important statistical descriptors

of that variable. Since density functions are monotonically increasing, the value of the largest distance D which represents the difference between pdf_1 and pdf_2 as they range from 0 to 1 can be used to infer the difference between the distributions.

While not as tightly defined, the same principle is extended to two dimensional pdfs and a similar statistic is generated and used to apply the (dis)similarity hypothesis. This generalization (2DK-S) was first presented by Peacock [69]. More recently, Fasano and Franceschini [24] proposed and implemented a variation of the work and it is this version which is selected for this work.

The two sample form of this test (2D2SK-S) permits the generation of a statistic that can indicate whether the difference between two behaviors is statistically significant. This approach has already been used to study data from Supernova 1978A [90] and also to study X-ray images of ROSAT sources in NCG6307 [55]. These two examples are both found in astronomy but they provide support for the belief that the 2D2SK-S test is applicable to behaviors in this manner.

3.3.2.3 Details of the 2D2SK-S Test

Using Fasano and Franceschini's implementation, the number of examples in each of the four quadrants around a given example (s_j, a_j) are calculated. These terms are the fraction of examples in these four quadrants $(s_j \leq s, a_j \leq a), (s_j \geq s, a_j \leq a), (s_j \leq s, a_j \geq a),$ and $(s_j \geq s, a_j \geq a)$. The 2D statistic is defined as the maximum difference D between the fractions for corresponding quadrants when ranging over all the examples in each behavior. In [55] researchers have identified some drawbacks that make this definition of cumulative distribution less stringent than the one dimensional case but results have shown that the statistic can still be used to provide a useful estimate.

In Table 4, the proportion of examples in each of the four quadrants about the point $(50, 12)$ are listed for each of the considered behaviors - wall following, wall

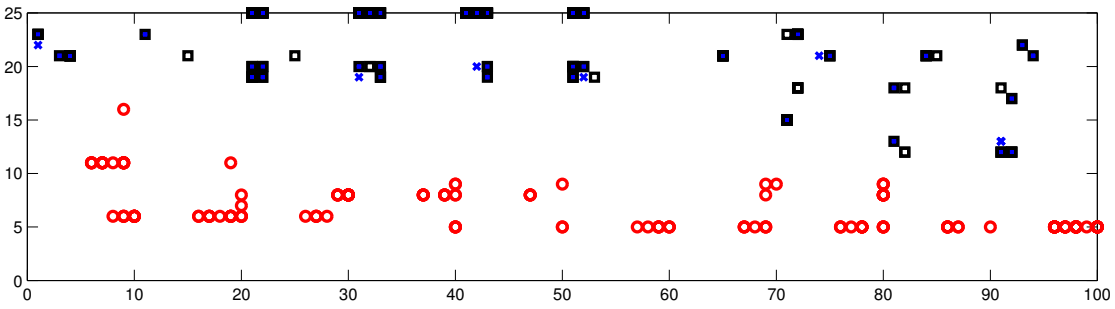


Figure 36: Examples from three behaviors superimposed on the same figure. Examples labeled as "Follow" in red circles, those labeled as "Avoid" in black squares, and those from the unlabeled "Behavior X" in blue x's.

Table 4: Distribution of examples (sensor and action state pairs) provided for three behaviors in the quadrants about point (50,12).

Behavior	Q1	Q2	Q3	Q4
Follow	0	0.0200	0.5000	0.4800
Avoid	0.3953	0.5581	0	0.0465
Behavior X	0.4130	0.5652	0	0.0217

avoidance and an unlabeled third behavior. At this investigation point, it is clear that there is a difference between the proportions listed for the un-labeled behavior, Behavior X, and those listed for the Follow behavior. The difference from the Avoid behavior is less significant, so evaluating these behaviors at this point $(50, 12)$ provides support that Behavior X is an example of the same behavior which the Avoid behavior approximates.

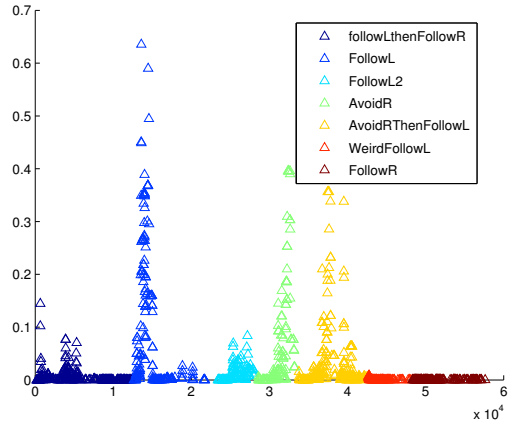
Fasano and Franceschini's algorithms are presented and implemented in C++ functions in [73] and it is this code which is applied in this work. As the test is slightly distribution dependent, the resultant probabilities are only estimates. It should be noted that this implementation does not consider all possible quadrants covering the example space. For each pair of behaviors considered, the algorithm iterates over the "points" in each of the distributions then averages the calculated distribution statistics to determine the statistic used for the test.

The result of the application of the 2D2SK-S test indicates that there is an estimated 0.9794 probability that Behavior X is an example drawn from the same behavior distribution function as the Avoid behavior. For completeness, the comparison for the Follow behavior yielded a probability estimate of 0.0000.

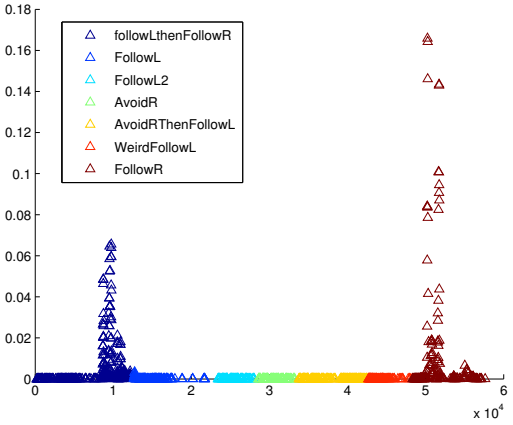
3.3.3 A View of the Data

To illustrate this data graphically, two behaviors are compared to a sequence of seven behaviors (See Figure 37). These behaviors included examples of wall following (both left and right sides), wall avoidance, and combinations of both of these capabilities. Two reference behaviors were used to show the how coherence was quantified. The first, used to generate the plots in Figure 37(a), was an example of wall following on the left side of the robot. The second, used for the data in Figure 37(b), was an example of wall following on the right side. In both cases, the base behavior was compared with a window from the sequence. Contiguous examples from each sequence were used but it is also possible to perform this exercise with a random selection from each of the candidate behaviors.

Each case that the behavior present in the sequence of behaviors is similar to the reference behavior, there is a strong "response" in probability (i.e. large probability values). As previously indicated, the guidelines state that the probability values less than 0.2 are associated with rejection of the NULL hypothesis. In practice, for some cases it is possible to lower this threshold. In [77] it was shown that the threshold could be set to 1/10 of the median probability and the result effectively characterized sets of example of behaviors. If this threshold is considered, then the data observed in Figure 37(b) would be able to be utilized (all the probability values are below 0.2). The effect of utilizing the median is that the most likely behaviors are able to be identified, and if all the behaviors are equally likely, then none are identified. For this metric, if a sliding window is used there will always be probability values that are



(a) Wall follow left.



(b) Wall follow right.

Figure 37: The probabilities of similarity generated for the listed reference sequence and the sequence of seven behaviors.

zero, so even in the case where a behavior is compared with itself, this threshold will be applicable.

3.3.4 What Information Can These Properties Provide?

At the base level, application of such a goodness of fit test permits behaviors to be differentiated. Only empirical observations are required to produce results in this case. While the use of a miracle distribution may not be ideal, it provides information where otherwise it would be difficult to acquire. The comparison of the representations of wall following and wall avoidance in the previous section is an example of the application of this concept.

The technique also enables a known behavior to be identified from a sequence of unknown examples. Again all that is required is an observation of the behavior. The advantage of being able to recognize behaviors that are not similar (i.e. not likely to be examples of the same target behavior) is that it becomes possible to filter out examples that should not be grouped together during the learning process. The old adage of garbage in garbage out is applicable when techniques are applied to "bad data sets". If the dataset contains examples of multiple behaviors than trying to fit these examples to a single model has been shown to produce poor results. Since reducing the load on the instructor in this interactive process a goal, this technique which holds the promise for reducing the burden of labeling provided instruction is thus of great value.

3.3.5 Alternative Strategies

Review of the literature indicates that Kullback-Leibler (KL) divergence and other probability distance metrics are frequently applied approaches to address the concerns of this section. These approaches develop a measure of difference between two probability distributions which is non-negative when there is a divergence between

Table 5: Entropy for the entire behavior observed in the following 5 examples

Behavior	Entropy
Follow left1	3.9288
Follow left2	3.4930
Follow left3	3.9372
Follow right	3.9822
Avoid	7.7474

the two, and zero only if the measures are the same. If the measure were a symmetric function of the terms and if it also satisfied the triangle inequality, it could be considered a true distance metric, however this is not the case.

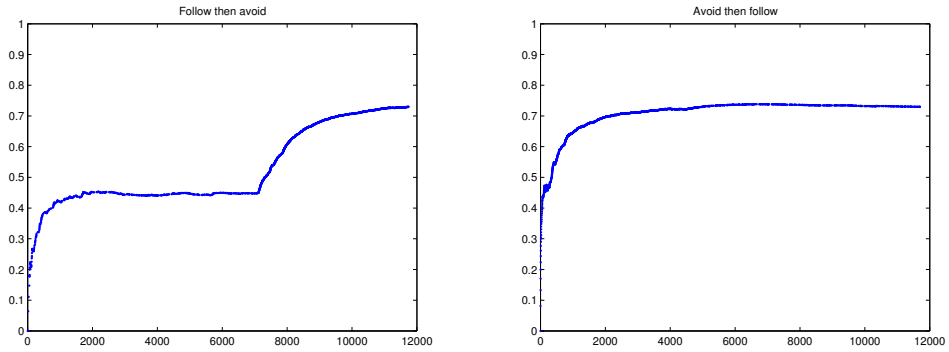
This failing is not however the largest challenge in applying these approaches to this task. Since these measures incorporate the properties of the whole behavior (in the case of KL the behavior's entropy), they cannot differentiate between probability distributions that have similar aggregate statistics. To illuminate this shortcoming the following example is presented.

Consider five examples of behaviors, demonstrated by a single instructor. In these cases interactive learning is not applied and these examples comprise the result of capturing sensing and action data from teleoperation. The first three examples are captured while the instructor demonstrates "Wall Following", while following the wall to the left of the robot. The fourth example was captured while the instructor demonstrated the same task, except this time the instructor followed the wall to the right of the robot. In the final example, the instructor roamed the arena, avoiding walls whenever they were encountered. The entropy of these tasks are presented in Table 5.

Different behaviors may indeed possess different entropies, however this difference is not a necessary property to identify that they are different. In Table 5, all three of the entropy values for the examples of wall follow left behavior differ. There is a large difference between these three and the entropy for the wall avoidance behavior,

however the difference for another behavior, wall follow right, is is not statistically significant. Since entropy is a function of the information states of the distribution, it is feasible and possible for different behaviors to have the same entropy. It is also possible for the same behavior to have different entropy values when demonstrated at different times or by different instructors. These challenges all result in the absolute value of entropy being of little consequence.

Based on Section 3.2.3, one may ask whether the change in entropy may be able to provide useful information, especially when considering a sequence of examples. An assessment of the data presented in Figure 38 quickly indicates the problem with such an approach. It seems that it is possible to identify a difference in how the entropy changes as examples from the avoid behavior are added to the examples from the follow behavior (See Figure 38(a)). This change is evidenced by a significant increase in entropy once examples from the avoid behavior are incorporated. When the sequence is changed by presenting the avoid behavior first (Figure 38(b)), such a change is no longer discernible.



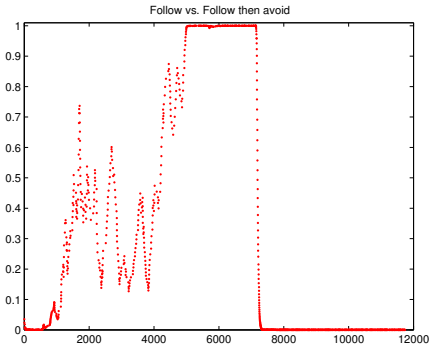
(a) Examples from the follow behavior then the avoid behavior.

(b) Examples from the avoid behavior then the follow behavior.

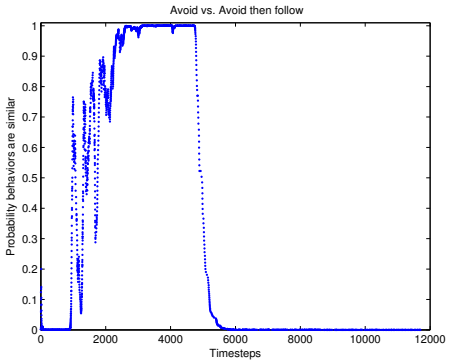
Figure 38: Entropy observed for two sequences of two behaviors.

When the 2D2SK-S test is applied however, the differences between behaviors becomes readily evident, regardless of the order in which examples are considered. As indicated by the data presented in Figure 39, the respective null hypotheses are

rejected as expected in both scenarios. In each case, when a different behavior is presented, the probability fell to zero.



(a) Examples from the follow behavior then the avoid behavior with a follow behavior as the reference behavior.



(b) Examples from the avoid behavior then the follow behavior with an avoid behavior as the reference behavior.

Figure 39: Probability that elements of the sequence as similar to the respective reference sequences.

CHAPTER IV

APPLICATIONS OF COHERENCE

In Chapter 3, three types of information were presented which could be generated during the process of interaction between the robotic student and the human teacher. These pieces of information are approximations of information that human students typically use when learning.

The first, classification error, provides a measure of how familiar the provided instruction is to the student. For example, when new terminology or new jargon is presented in a classroom, human students are initially unaware of their meaning. With continued use however such language is incorporated into the student's vocabulary. This process is equivalent to the robotic student's reduction in classification error as they learn to classify the instruction.

The second type of information, behavior entropy, provides a measure of how coherent the behavior is observed to be. As is the case in many human classroom settings, if teachers are inconsistent, the answers that they provide lead to uncertainty during the learning process.

The third type of information, statistical similarity, also targets inconsistency. This measure captures the relationship between two or more observed behaviors. This measure approximates the reasoning process that a student undergoes when they recognize that their teacher is repeating an act with which they are familiar. It is the same process which is applied when a student rises to challenge the teacher's assertion that this method has been previously presented. For the robotic student, statistical similarity based on goodness of fit tests can be used in like manner. While not as sophisticated as the reasoning process for the human student, this property

can be readily derived with information available to the robot.

In this section the purpose is to show how each of these can be applied to aid the process of interactive learning with the human teacher. To show how this information can be applied by a robotic student, the following results are considered. Each of these results will show how the robotic student can generate information without direct involvement from their teacher and how this information can help the teacher teach them.

4.1 Identifying When to Stop Learning

At the end of sections 3.1 and 3.2, two estimates of current learning states are presented. These estimates can be used to indicate when it is time to stop learning. If the estimate indicates that learning is progressively increasing, continued learning should provide additional benefits.

In time constrained situations, where learning cannot continue indefinitely, the estimate can be used to predict when a specific milestone will be attained, or how many examples it will take to improve by a specific margin. If the estimate indicates that the performance is not improving or even degrading with continued instruction this can be used as a trigger to terminate the learning process. This last situation, where instruction does not help performance, is quite possible in situations where bad instruction is provided.

Data from twelve teaching sessions were used to study how to apply this information. The sessions were comprised of four groups of three runs. The first two groups were from the two scenarios presented in Chapter 3 as scenario A and scenario B. A third scenario, labeled C is introduced in which additional random noise is added to each actuation channel. All of these additive terms are independent identically distributed Gaussian with the mean 5 and variance 2. The range of each of the actuation channels is 10. Scenario D is attained by performing learning under the conditions of

both Scenario B and Scenario C. This means both that the instruction provided and the associated sensor data are corrupted with noise on some channels.

There are two approaches that can be utilized to determine when to stop. The first is to define a stopping rule based on local trends in the estimate. This approach monitors changes in the current estimate of the performance and how it relates to the previously calculated estimates. If the current estimate indicates a trend that does not provide confidence in future examples, then it can be used as justification for ceasing to acquire new examples.

The second approach involves using the estimates to generate predictions for future estimates. In this case these predictions are based on the assumption that robotic students possess learning curves akin to those of human students. By generating a learning curve based on the student's estimate of it's learning progress, it can predict when it will attain a given performance level.

Both of these approaches exist under the assumption that the instruction approach under which examples are provided does not change significantly over the course of interaction. While an instructor can be expected to become better able to provide instruction with additional practice, it is assumed that they remain faithful to do things such as provide the best instruction they can whenever it is needed.

4.1.1 Stopping Based on Local Trends in Generated Estimates

The stopping rule that will be evaluated is the following: For performance estimates E_j and E_m where $j > m$, stop at estimate E_j if $E_{j+1} > E_j$ and $E_{m+1} > E_m$. This rule has the result of stopping after the second degradation in performance (as estimated by the student), during the learning process. For the situation presented in Figure 40, the stopping rule triggers at the fourth period (time step 5200). This stopping rule is applied to data gathered during each of the conditions indicated. After learning is concluded, the learned controllers are applied to evaluate the performance. For each

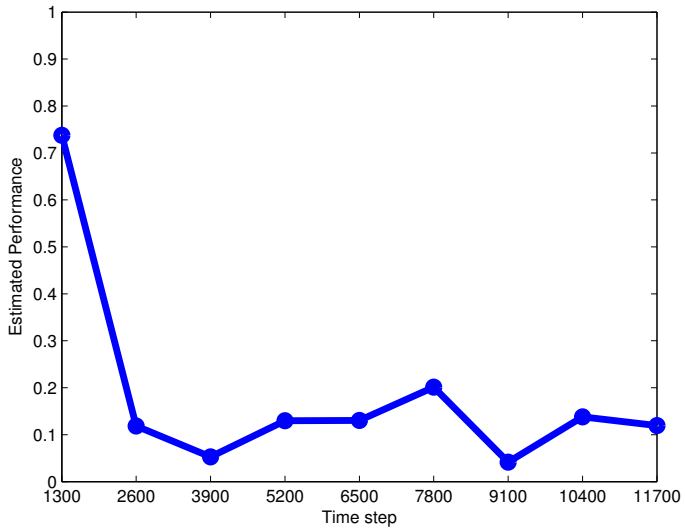
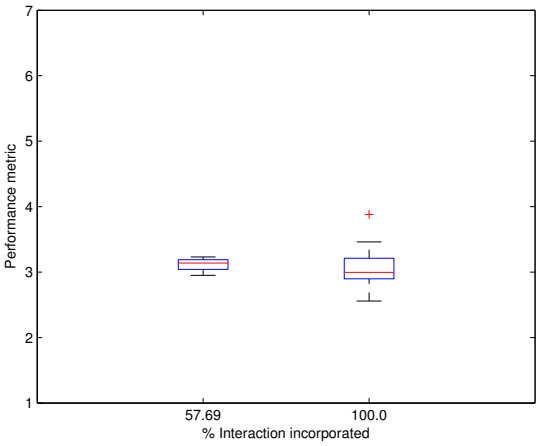


Figure 40: The estimate of performance generated using the information extracted from entropy for one the sessions where learning occurred under the conditions of scenario D.

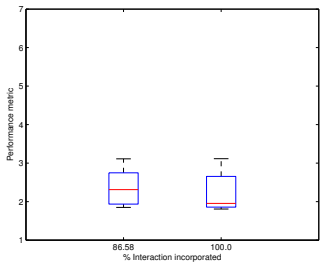
controller, the performance is evaluated twenty times.

The plots in Figure 41 present the comparative base plots between learning with all the data, and learning with the reduced data set, for each of the cases where the trigger was applied. These plots show that the coherence metric was generally able to provide insight about the actual performance of the student as they learned. In most cases the difference between the coherence indicated stopping point was not statistically different from the cases where all the data was used. These results indicate that coherence can be used to equip the student to provide useful feedback to their instructors. At the trigger point for the stopping rule, the student can indicate that they believe that the current learning state is a good stopping point.

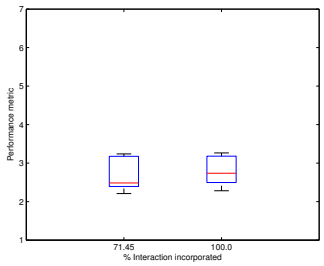
Table 6 shows the results of the application of the stopping point in the cases where the stopping point occurred prior to the end of the interaction process. Previous figures have shown that in only one case the performance difference was statistically significant, and in this case less than half of the examples were applied. The triggering of the stopping rule implies that instead of continuing this session, beginning another



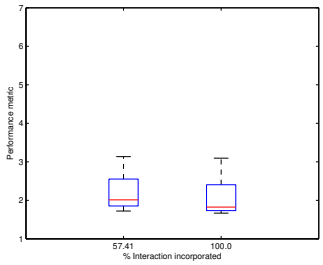
(a) Case B1.



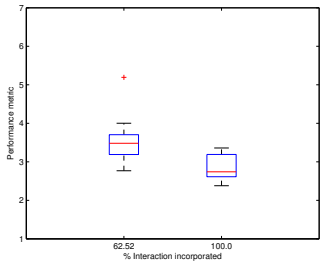
(b) Case C1.



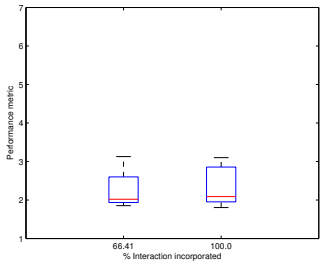
(c) Case D1.



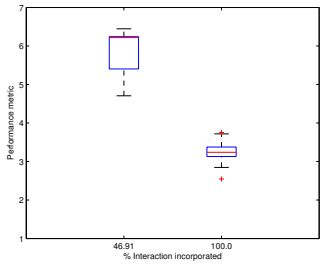
(d) Case C2.



(e) Case D2.



(f) Case C3.

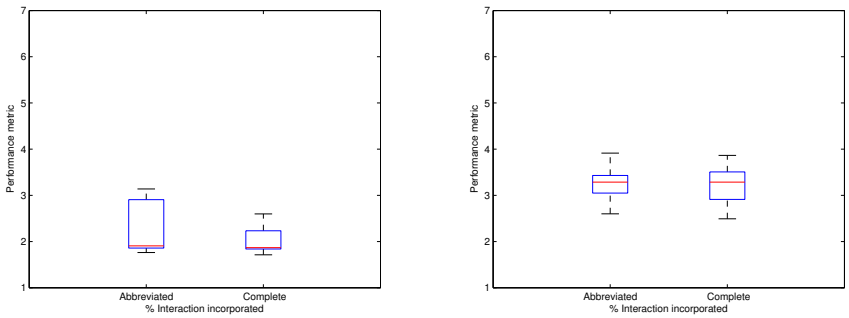


(g) Case D3.

Figure 41: Box plots for the performance in the cases where the training set was truncated and the results when all the data was used for all the data.

Table 6: Effect of applying stopping rule

Scenario	Median Performance	%Median Performance lost	%Examples saved
B	2.778	-7.26	42.31
C	2.311	18.30	13.42
C	2.011	10.26	42.59
C	2.020	-3.28	33.59
D	2.485	-9.24	28.55
D	3.482	27.11	37.48
D	6.218	92.01	53.09



(a) Scenario C. (b) Scenario D.

Figure 42: Box plots for the performance of learning with data aggregated from all the examples in each scenario. The plots show the performance when data was aggregated both when the stopping rule was triggered, as well as when all the instruction was incorporated.

would have yielded better results. To confirm this implication, the results of grouping multiple sessions gathered under the same conditions (scenarios) are considered.

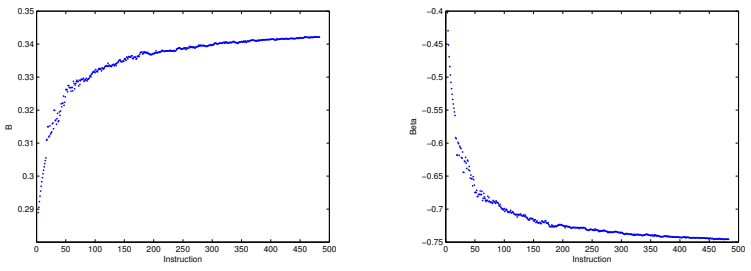
Figure 42 presents the box plots for the two scenarios in which the stopping rule was triggered for all the sessions. There is no basis, statistically speaking, for continuing the learning process in each of these six cases (three sessions for each of scenarios C and D). This claim is supported by the fact that when the examples are aggregated, there is no additional benefit provided after the stopping rule was triggered. It should be noted that this benefit was provided without knowledge of the target task or even a priori information on useful perceptual states in the instruction provided. Had an approach like this been applied, the student would have saved their teachers significant time contributions, and suffered no degradation in performance.

4.1.2 Stopping Based on Predictions of Performance

Again continuing the theme of what the student can recognize, it is also possible to connect the estimates of past and current learning states to generate predictions of future learning states. In [76] it was proposed that robotic students possess learning curves akin to those of human students. With the knowledge that the learning curve followed the power law of learning (See Section 2.3), it is possible to apply regression techniques to determine the parameters of these curves. Nonlinear least-squares regression, implemented with the SOM [45], is applied in all the cases considered in this section.

It should be noted that if the student is able to generate its learning curve without the direct aid of its teacher, the not only can it determine when to trigger a stopping rule, but it can also provide useful information back to its instructor. Without having to communicate details about the task, or the process of teaching, the instructor can be provided with the assessment (or useful parts thereof) generated by the student, and thus make better informed decision about teaching. This type of information is often present in human learning scenarios and is missing from the human-robot study.

In Figure 43 the predicted values for B and β are presented for the interactive learning session under scenario A. From this figure it can be seen that after receiving 180 instruction items, the estimates of the learning parameters defines the learning



(a) Learning range. (b) Learning rate.

Figure 43: Learning curve parameters generating for learning during scenario A1. These parameters are incrementally estimated based on the instruction received.

curve presented in (19). At this point in the learning process, the student, with the use of the information provided in this curve would be able to infer that 66% more examples would provide a performance improvement near 40%.

$$\hat{P(N)} = 0.3359 * N^{-.7091} \tag{19}$$

For the twelve sets of data considered, and based on the estimated learning curve after receiving 180 instruction, the prediction of the performance after 300 instructions was calculated. The error between these predictions and the values measured when that learning was attained are presented in Figure 44. This box plot shows that the median error is slightly above zero but more importantly, zero falls between the upper and lower quartiles. For the data considered, the performance prediction was not far off the actual. This result suggest again that it is possible for the student to predict when it would be able to attain a specific performance level based on the estimates it is able to generate. The estimate can also be used to infer when it has attained the target performance, and thus can be used to trigger the stopping rule. It should be noted again that these predictions are based solely on the nature of the provided instruction. The application of the prediction in this manner is also based on the number of examples (N) provided and not the time(steps) elapsed during

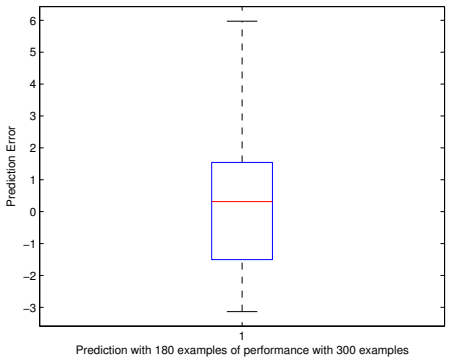


Figure 44: The box plot for the error in the prediction of the performance after 300 examples. For this data, the prediction was made after incorporating 180 examples.

interaction. This this outlines another area where the assumption that the teacher remains consistent in their teaching approach influences the robot's ability to assess its performance.

4.2 Identifying Different Examples

In the previous section, approaches were shown which could enable the student to assist the instructor in determining when to stop a learning session. In this section the question of what to do with the sets of acquired instruction is addressed. In Section 4.1.1 it was shown that the student could aggregate sessions to learn the task. The motivation was not presented there, however it is sometime useful to incorporate multiples examples of a behavior to better learn a task. Multiple distinct examples of the same task often help to solidify "understanding" of how the task ought to be performed. If "bad" examples of a task, or examples of different types of task are included in the training sets, this can result in poor learning outcomes.

In the context of this research, it cannot be expected that the student can recognize what "bad" instruction is. Such a value judgment is beyond the capability of the student. What is not beyond reach however, is the ability of the student to recognize what can be loosely termed "similar" instruction. If a teacher is teaching a specific behavior, the relationship between what is done and when it is done should be consistent over the various learning sessions. While variation can be expected, gross contradiction in the provided instruction would suggest that there is something other than the ideal learning circumstance afoot.

The approach presented in Section 3.3 provides a technique to identify different types of instruction. It is useful that this approach is based only on the sets of provided instruction, since the student can apply it without need for any other auxiliary resources. This degree of autonomy should permit the student to effectively learn the tasks without increasing the responsibilities of their teachers.

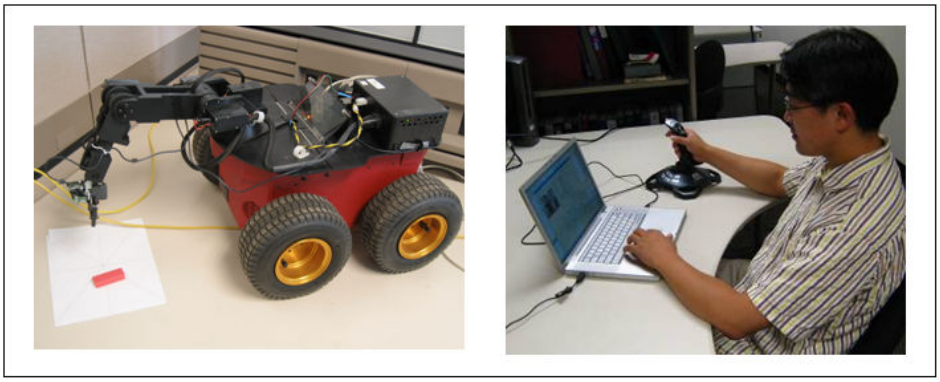


Figure 45: Teleoperative manipulation system using Pioneer3AT and Pioneer Arm.

4.2.1 Manipulation Data

The data used for this section was gathered using the rig shown in Figure 45. This rig is composed of a Pioneer3AT mobile robot, the 5-DOF Pioneer Arm, a USB camera, a computer (laptop) and joystick to run the control code. The USB camera is mounted just over the gripper so that the workspace can be viewed by the instructor. The joystick was used to generate three control channels which were combined to control the arm Two of the joystick axes were mapped to control the x-axis and y-axis of arm's end-effector, and the third is used to control z-axis. The window size of the mapping between joystick and arm's workspace adaptively changes as it approaches the object.

Multiple demonstrations of two tasks were considered in this section. The aim in all cases was to teach the robotic system to reach down and grasp an object which was initially in the view of the eye-in-hand camera. Demonstrative trajectories were captured while a user picked up an object from distinct locations on the surface. The overall training set would feature an object on a planar surface in a 15cm by 10cm area.

The second task was similar to the first except that the start position of the end effector was initialized near to the target object. This task posed slightly different challenges since each trajectory would contain a fewer number of items and errors

early in the motion sequence could have greater impact on keeping the object in the portion of the workspace where it was visible to the camera.

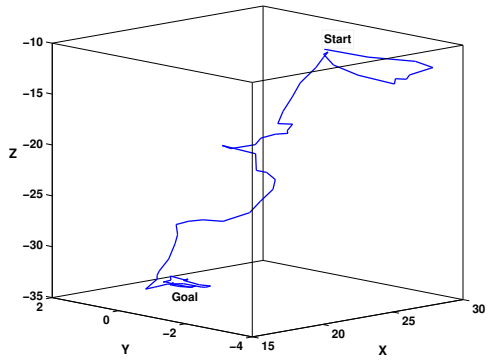


Figure 46: End effector trajectory of a good example.

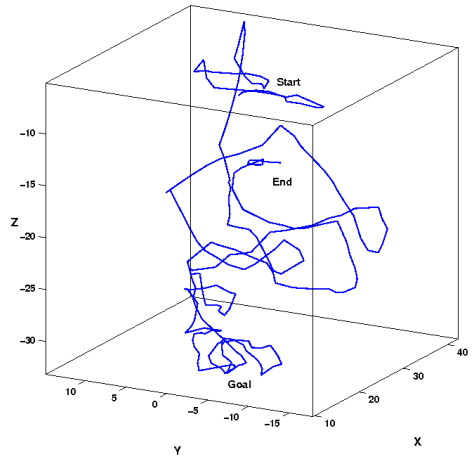


Figure 47: End effector trajectory of a bad example.

For each behavior, the expert user provided examples sets which they indicated as good or bad. Nine sets of good examples and three sets of bad examples were provided by the expert user. Examples of the end effector trajectories for one good and one bad example is shown in Figures 46 & 47 respectively. For conciseness, the labels of the training set gathered are presented in Table 7.

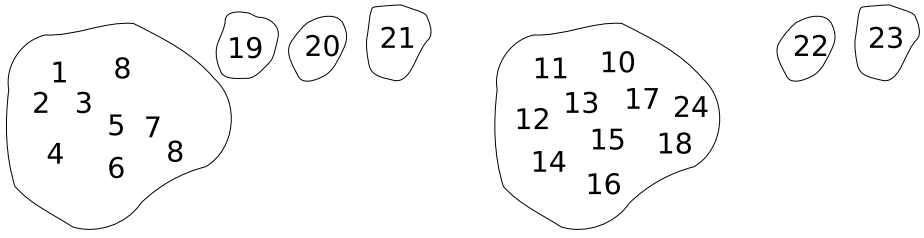
The results of applying this approach to evaluate the 24 trajectories considered in this work are presented in the candidate sets shown in Figure 48. Figure 48(a) shows

Table 7: Expert labeled trajectories for each behavior.

Label	Behavior	
	1	2
Good	1-9	10-18
Bad	19-21	22-24

the behavior candidate sets uncovered for Behavior 1. This figure groups trajectories 1-9 into a single candidate set and trajectories 19, 20, and 21 into separate candidate sets. Such groupings provide support for training a single ANN with trajectories 1-9, separately from the other examples. It is interesting to note that these were all the examples of good behaviors. Without prior knowledge about the training set, the approach permitted the good behaviors to be grouped into the same behavior candidate set. It is also significant that each of the bad trajectories is relegated to its own candidate set. This is not alarming since it was unlikely that each of these trajectories would be distorted in the same manner. A high degree of connectivity was observed in the graph which captured the relationship between trajectories 1-9. This indicates to some degree the confidence that each of these trajectories are examples of the same behavior.

A similar story is told in Figure 48(b). In this figure the trajectories are presented for Behavior 2. The behavior candidate sets group trajectories 10-18, and 24 into a single candidate sets and trajectories 22 and 23 are relegated into their own candidate sets. Unlike the case with the examples of Behavior 1, a single behavior from the user



(a) Behavior 1. (b) Behavior 2.

Figure 48: Grouping of trajectories for two behaviors.

Table 8: Groupings utilized to evaluate the efficacy of the application of coherence to this data.

Behavior	Grouping	Example sets
1	I	1-9
	II	1-9, 19-21
2	III	10-18
	IV	10-18, 22-24
	V	10-18, 24

defined bad set is included with the good trajectories.

To test the effects of the candidate set groupings, the groupings sets of trajectories presented in Table 4.2.1 are considered as training sets for ANNs. Comparing the differences between learning from groupings I and II will show the expected benefit of learning from the CBF suggested examples of the behavior over learning from the entire set of examples of that behavior.

None of the candidate groupings for Behavior 2 are equivalent to perfect knowledge of good examples. This provides an opportunity to demonstrate what is likely to be useful information to evaluate this approach. Since an expert user presented the trajectories for III, it is expected that the ANN trained with that set would outperform the sets IV and V. The outcome of a successful test of this approach would show that the performance of a network trained with V exceeds that of one trained with IV. Such an outcome would be better only if the performance after learning with V is equivalent to that of learning with III. When this data is used to generate neuro controllers, it was noted that the grouping (I and V) led to neuro controllers which converged more quickly and produced more accurate results than if all the data (II and IV) was utilized [77]. This is another example of how coherence can lead to generating better results faster, without a great deal of up front information about the considered tasks.

4.2.2 Activity Recognition Data

To provide another aspect of the potential afforded by using the concept of coherence, video clips of subjects performing different arm exercises, of varying degree profiles, in seated and standing position, and with the left and right arm, was collected (Figure 49). The three different arm exercises considered were:

- Shoulder Flexion - Raise arm to point to ceiling, keeping elbows straight
- Shoulder Abduction - Raise arm out to shoulder level, keeping elbow straight
- Shoulder Rotation - Keep elbow in place and slide forearm back and forth

These exercised were selected from the American Academy of Orthopedic Surgeons, Shoulder Surgery Exercise Guide [2].

Figure 49: Example sequence of images captured during observation. 180 degree left shoulder abduction (top), 90 degree left shoulder abduction (middle), and left shoulder rotation (bottom).

Unlike the previous experimental scenarios, an actuation channel is not explicitly available. Since it is known that periodic behaviors are being observed, it was possible to extract the period and use it to generate the needed channel. Application of such a technique simply serves to fill in information which would have been provided by

Table 9: Test behaviors used.

Test	Exercise	Description (camera view)
1	#1	180 Right Shoulder Abduction Seated Position (front)
2	#1	180 Right Shoulder Abduction Seated Position (front)
3	Incorrect	180 Left Shoulder Abduction Seated Position (front)
4	#1	180 Right Shoulder Abduction Standing (left side)
5	#1	180 Right Shoulder Abduction Standing (right side)
6	#2	90 Right Shoulder Abduction Seated Position (front)
7	#2	90 Right Shoulder Abduction Seated Position (front)
8	Incorrect	Off Angle Right Shoulder Abduction Seated Position (front)
9	Incorrect	Off Angle Right Shoulder Abduction Seated Position (front)
10	#3	Sup. Right Shoulder Rotation Seated Position (front)
11	#3	Sup. Right Shoulder Rotation Seated Position (front)

the physical therapist as they guided the subject during the exercise. To evaluate the benefit of the use of coherence to group observed behaviors, the following test sequence was utilized.

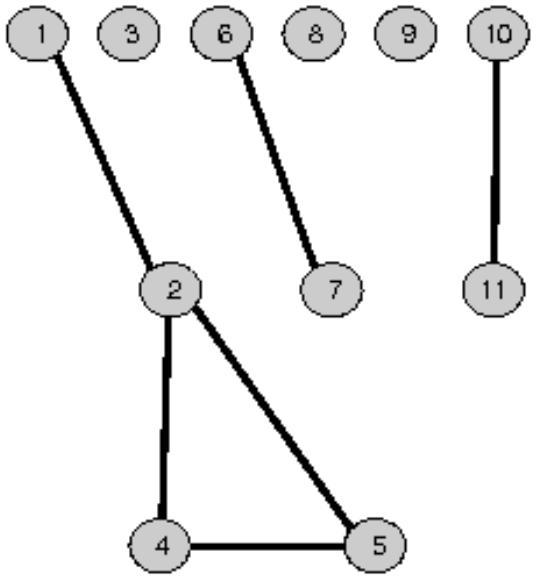


Figure 50: Graphical representation of the adjancy matrix. Nodes (exercises) which are connected are those which are identified as similar.

The resulting categorization of exercises grouped Exercise 1, 2, 4, and 5 into one class. Exercises 6 and 7 were grouped into one class. Exercises 10 and 11 were

Table 10: Results generated.

Test	Exercise	Classified with exercise
1	#1	#1
2	#1	#1
3	Incorrect	None
4	#1	#1
5	#1	#1
6	#2	#2
7	#2	#2
8	Incorrect	None
9	Incorrect	None
10	#3	#3
11	#3	#3

grouped into one class, and Exercises 3, 8, and 9 each into separate classes. The resulting classification scheme is summarized as shown in Table 10

These results again show support for the use of this approach in identifying dissimilar behaviors. This awkward wording is used since the test does not indicate which similarity, but lack of evidence of a significant difference. This is a subtle but important distinction.

4.2.3 Recognizing When to Change Approaches

There is one more area where the student can apply this information in a useful manner. In the previous cases, the information was applied to group examples when given M examples of N behaviors where $N \leq M$. We now consider the case where one behavior is being presented and examples from a different (distinct) behavior are introduced. In Section 3.3 when this test was introduced, it was applied to what was known to be two different sets of examples. These examples could quite possibly been two different behavior, but the beginning and the ends of each example were known. Now we consider the case where the student just receives a stream of examples. It is up to them to recognize that the examples of the first behavior are no longer being provided by the instructor. Without a prompt from the instructor they should either

issue a query or label the more recent examples as those for a different behavior.

The data used in this section was collected in the environment shown in Figure 52. Three basic behaviors were demonstrated in each case the robot was taught a navigation task that it accomplished using camera data and ranger data as its sensing modalities. The camera data used for interactive learning (and for the robot's operation), was provided from an on board camera attached to the robot's chassis. This camera provided a forward facing view of the environment around the robot within a 60 degree field of view. The image in Figure 51 is one of the images presented to the robot during interactive learning.

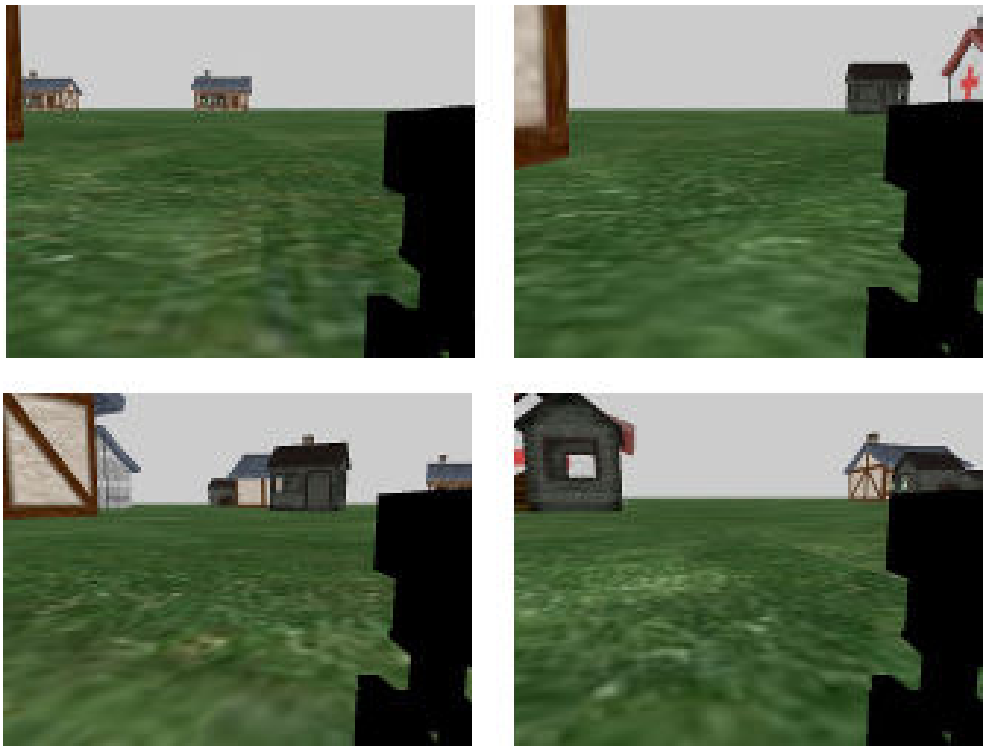


Figure 51: Assorted views from different locations around the simulated environment from the on board camera. These types of views were used both by the student when operating autonomously, as well as by the human teacher to determine what instruction to provide.

In the first of the three behaviors considered, patrolling clockwise, the robot is taught to navigate around the village and maintain a path that keeps it along the outer perimeter of the 11 building complex. This behavior was taught to the robot

twice and is labeled behavior B1 and B3.

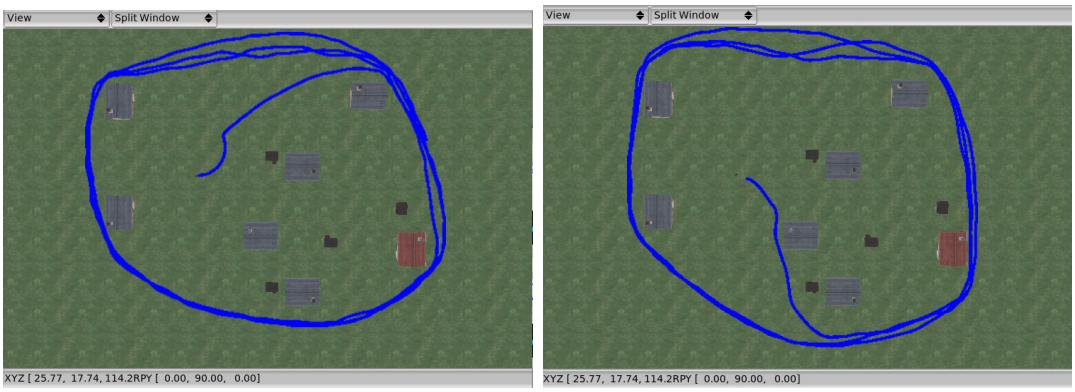
The next behavior is similar to the first. Instead of traversing the environment in a clock wise manner, the robot is taught to travel in a counter clock wise direction. The path shown in Figure 52(b) was captured while this behavior was being taught. Unlike Figure 52(a), when learning B2, the robot keeps the building to its left as it navigates through the environment. Behavior B4 was collected by permitting the instructor to teleoperate while demonstrating the same behavior. The only difference between behavior B4 and B2 is that the robot did not learn interactively.

The final behavior is a bit eccentric and is truly included as a counter example to the behaviors included thus far. In this behavior, the robot is taught to navigate through the village in an erratic manner. There are times during instruction where the instructor drives through buildings in the village. This is possible since many of the objects in the environment do not possess solid bounding boxes.

For each behavior, two hundred samples were randomly selected and compared against two hundred samples from a window of 800 samples from the sequences of examples. As new samples are provided in the sequences, the window was incrementally advanced and a new set of 200 samples acquired. Applying the methods presented in Section 3.3, the metrics generated are presented in Figure 53.

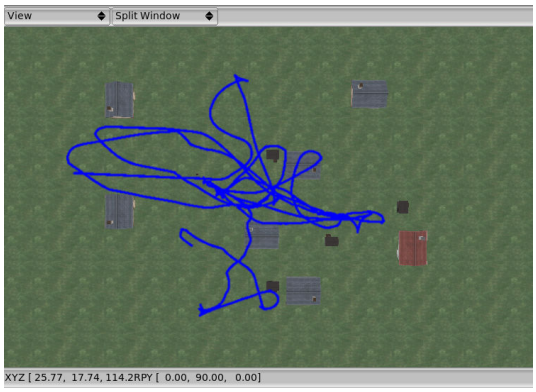
Inspection of Figure 53(a) shows that, as expected, examples from B1 are very similar to the reference behavior (which is B1). As examples from B2 are introduced, the similarity probability decreases, This suggests that samples from B2 are not like those in B1. When samples of B3 begin to be incorporated, the probability increases once more as expected since B3 is another instance of the same behavior. A similar relationship between these three behaviors is observed when considering Figure 53(c). This figure presents the same data using behavior B3 as the reference behavior.

If knowledge about the existence of behaviors B1, B2 or B3 did not exist, the data shown in these figures provides insight that there is enough information available



(a) Path for B1.

(b) Path for B2.



(c) Path for B5.

Figure 52: Paths demonstrated as the robot is taught interactively.

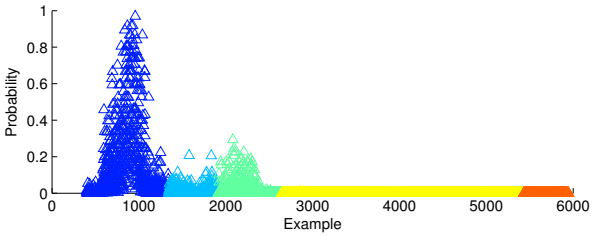
to make and educated guess about whether examples of a different behavior are being considered. In cases where the metric initially indicates a favorable comparison between the reference behavior and the current window of the training sequence, then the comparison becomes less favorable, this is a strong indication that there is a change in the examples.

To show how this metric could be converted, Figure 54 presents the cumulative sum of the data in Figure 53(c). There are two sections of this graph where there is a linear increase. These two sections are those where similar behavior are recognized. There are other areas where the slope is non-zero. In these areas the sequences are recognized to have a small degree of similarity with the reference behavior, but by applying a slope threshold, some of those regions can be eliminated.

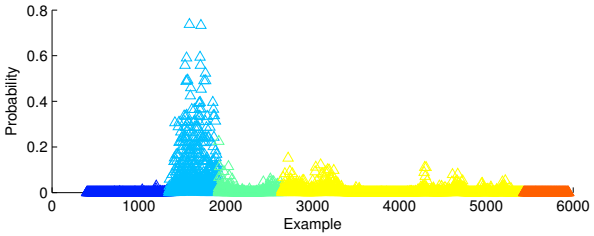
4.3 Summary

In the applications of coherence considered, an astute reader may recognize that there is a possibly glaring omission in the information that the student can recognize. The student can recognize when to stop learning, how to group examples in a beneficial manner and even when to classify new examples as a new behavior. What has not been discussed however, is the student's ability to recognize that it is acceptable to start learning from a given instructor. It seems plausible that while using the provided tools the student could recognize that an instructor is highly inconsistent or is providing contradictory instruction. If this can be recognized then it would also appear plausible that the student could discount instruction deemed to be incoherent, until the teacher or the teaching they provide improves.

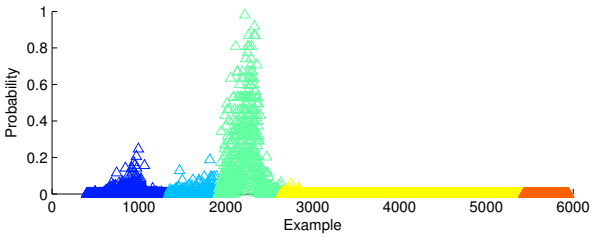
While such an approach seems plausible, it is not endorsed since it does not respect the role of the instructor as the primary source of instruction. This scenario is however not ignored. By observing multiple sets of examples of a given behavior, if there is initially horrible instruction provided, it is likely to be grouped separately from subsequent instruction unless there is no improvement over time.



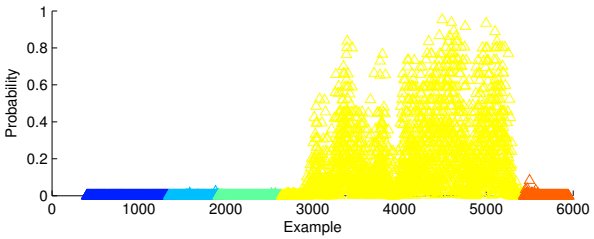
(a) Comparison with B1.



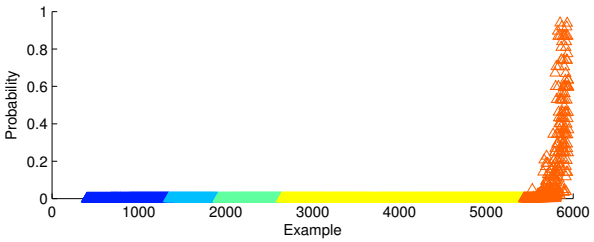
(b) Comparison with B2.



(c) Comparison with B3.



(d) Comparison with B4.



(e) Comparison with B5.

Figure 53: Probability that behaviors will be grouped together. The coloring is adjusted (advanced by half the window size) to accommodate for start up delay. The sequence of five examples are presented sequentially. Each different color indicates example for a different behavior. Strong response indicates behaviors that are identified as similar.

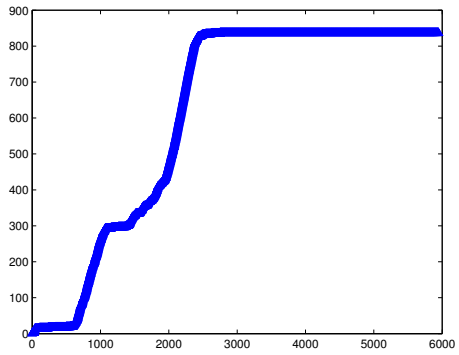


Figure 54: Cumulative sum of the probabilities presented in Figure 53(c).

CHAPTER V

CONCLUSIONS

During the generation of the proposal for this research, there were many ideas and avenues considered. Due to the experimental nature of robotics research, there were many opportunities for challenges provided by the required systems integration, usage of developmental open source code, and the always unpredictable hardware challenges associated with the use of mechanical devices. Other challenges included limitation of realizable goals contributed by restricted workspaces of actuators (and actuator chains), variable and undocumented operation rates for devices like cameras, and inability to fully access data and functionality of useful robotics and graphics repositories.

These challenges did not make the intended research any less difficult, and in spite of them, interesting results and understanding have been acquired. It is our hope that the included chapters have helped to convey this information and now, some of the key points are summarized.

For social creatures, like humans, communication is a valuable component of interaction. Currently, beyond explicit query, there are few avenues which can enable humans to get useful feedback. It was proposed that the instruction acquired, either incrementally or interactively, likely contained information which could be extracted by the robotic student. Further, it was advanced that this information could be extracted in a manner provided by the robot designer, without adding additional requirements or additional responsibilities on the robot operator (i.e. the teacher).

Some of the information was uncovered and in this work each piece is based on coherence, a property which appears critical for learning in this manner. Coherence

was defines as the term which quantifies the consistency between two types of related information. In this work, coherence was observed to have three forms: between sensing and action, between sets of examples acquired while learning a single behavior, and between distinct sets of examples.

Using these three forms of coherence, we were able to demonstrate how a robot could be equipped to assess its performance *while* learning, generate predictions about its future performance with additional instruction, and even identify conditions under which it could suggest that it should terminate the learning process. We have also demonstrated how the robot could identify properties of the provided instruction which indicate useful qualities about the instructor.

At the conclusion of this work, there are many interesting avenues which still remain to be explored. For example, studying learning while enriching the sensing and actuation modalities was successfully accomplished, exploratory work expanding the dimension of actuation by three orders of magnitude has been implemented but not fully studied. Another avenue ripe for study in Human-Robot Interaction would explore how robots can provide feedback in the most effective manner for their human teachers. While this work has shown how to grant the robotic student a voice, how they communicate with their teachers is as important a concern.

The scope of this work did not include multiple robotic agents, but extending the study to teams (or systems) of robots learning the same general environment would provide another rich next step. Such a study of the dynamics of learning in this "classroom" setting, where not only the teacher, but other students can provide instruction, is an avenue of interdisciplinary research which should be quite fruitful.

Finally, after individual behaviors are learned, it would be of interest to apply a similar method of interactive learning to learn how to compose these behaviors into sequences of "meta-behaviors". This extension could utilize a framework where skills are percolated from the bottom up and conceptual plans are filtered from the top

down, and where the two meet, new functionality would be realized.

These are a few of the thoughts on potential next steps. It is our hope that many of them are realized to advance the state of the art, and also to transform the way of life in even the smallest manner for the newest robot owners and operators.

APPENDIX A

GENERATING THE IMAGE SIGNATURE

Inspired by the work presented by Ng [57, 84], where it was leveraged to facilitate monocular depth perception, the following texture-based signature is generated from the energy uncovered by different types of filters. The core of this approach is convolution, specifically the two dimensional form shown in Equation 20. This equation defines the convolution of two terms, x and h, where x is the data (in this case an image), and h is the convolution kernel (or mask).

$$y(m,n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} x(m+i, n+j)h(i,j) \tag{20}$$

The kernels used are presented in Equations 22-35. The first nine of these kernels are from the set of 3x3 Laws' masks.The remaining kernels were selected by a trial an error approach, and no claim is made that they provide the most complete set of kernels.

These 15 kernels are applied to channels from the target images, specifically the three channels from the YCbCr color space representation of the image. In this color space Y (sometimes listed as Y') is the luma component (analogous to the the brightness), Cr and Cb are the red and blue differences of the chroma components.

To generate the signature, kernel h_1, is first applied to the Cr component, then to Cb component. After this step, all the filters are applied to the Y component, generating a total of 17 sets of results. In each of these results, the image is partitioned into 16 equally sized regions and the mean of the values in each of the regions is calculated. These means, 272 values, define the image signature for the image.

$$h_1 = [-1, 0, 1; -2, 0, 2; -1, 0, 1] \tag{21}$$

$$h_2 = [-1, 2, -1; -2, 4, -2; -1, 2, -1] \tag{22}$$

$$h_3 = [1, 0, -1; 0, 0, 0; -1, 0, 1] \tag{23}$$

$$h_4 = [-1, -2, -1; 0, 0, 0; 1, 2, 1] \tag{24}$$

$$h_5 = [1, -2, 1; 0, 0, 0; 1, -2, 1] \tag{25}$$

$$h_6 = [1, -2, 1; -2, 4, -2; 1, -2, 1] \tag{26}$$

$$h_7 = [-1, -2, -1; 2, 4, 2; -1, -2, -1] \tag{27}$$

$$h_8 = [1, 0, -1; -2, 0, 2; 1, 0, -1] \tag{28}$$

$$h_9 = [1, 2, 1; 2, 4, 2; 1, 2, 1] \tag{29}$$

$$h_{10} = [-1, 0, 0; -2, 0, 0; -1, 1, 1] \tag{30}$$

$$h_{11} = [-0, 2, -0; -2, 0, -2; -0, 2, -0] \tag{31}$$

$$h_{12} = [1, 0, 1; 2, 0, 2; 1, 0, 1] \tag{32}$$

$$h_{13} = [-2, 2, -2; -2, 0, -2; -0, 0, -0] \tag{33}$$

$$h_{14} = [1, 3, 1; 2, 0, -2; 1, 3, 1] \tag{34}$$

$$h_{15} = [-1, -2, -1; -2, -4, -2; -1, -2, -1] \tag{35}$$

Algorithm 2 presents the approach used to generate a portion of the image signature. As indicated, this algorithm is applied 17 times and since convolution is a pixel by pixel the image size directly impacts the processing time. In this work, the resolution of the images used is 640x480.

Algorithm 2 Process of generating portion of signature from current filter

```
select current kernel
tmp := convolution of image with kernel
for j = 0 to numblks-1 do
  for j = 0 to numblks-1 do
    blkIndex = base + numblks*i + j;
    for ii = 0 to blkWidth-1 do
      for jj = 0 to blkHeight-1 do
        pindex := j*blkHeight*imageWidth+i*blkWidth + jj*blkWidth + ii
        signature[blkIndex]:= value[blkIndex] + tmp[pindex]
      end for
    end for
  end for
end for
```

REFERENCES

[1] Alavarez-Ramirez, J., Cervantes, I., and Bautista, R., "Robust pid control for robots manipulators with elastic joints," in *Proc. IEEE International Conference on Control Applications*, (Mexico City, Mexico), pp. 542–547, Sept. 2001.

[2] American Academy of Orthopedic Surgeons, "Shoulder surgery exercise guide." http://orthoinfo.aaos.org/topic.cfm?topic=A00067: Last accessed January 2009.

[3] Arkin, R. C., *An Behavior-based Robotics*. Cambridge, MA, USA: MIT Press, 1998.

[4] Ashlock, D. and Freeman, J., "A pure finite state baseline for tartarus," in *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, (La Jolla Marriott Hotel La Jolla, California, USA), pp. 1223–1230, IEEE Press, 6-9 July 2000.

[5] Barto, A. G. and Dietterich, T. G., "Reinforcement learning and its relationship to supervised learning," *Journal of Intelligent and Fuzzy Systems*, pp. 47–64, 2004.

[6] Bekey, G., *Autonomous Robotics*. The MIT Press, Jun 2005.

[7] Bentivegna, D. C. and Atkeson, C. G., "Learning how to behave from observing others," in *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior. Workshop on Motor Control in Humans and Robots: on the interplay of real brains and artificial devices* (Hallam, B., Floreano, D., Hallam, J., Hayes, G., and Meyer, J.-A., eds.), pp. 272–281, Cambridge, MA: MIT Press-Bradford Books, 2002.

[8] Bentivegna, D. C., Atkeson, C. G., and Cheng, G., "Learning from observation and practice using primitives," in *AAAI Fall Symposium Series 2004: Workshop on Real-life Reinforcement Learning*, 2004.

[9] Berman, S., "Human free will in anselm and descartes," *The Saint Anselm Journal*, pp. 1–9, 2004.

[10] Blank, D., Kumar, D., Meeden, L., and Marshall, J. B., "Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture," *Cybernetics and Systems*, vol. 36, no. 2, pp. 125–150, 2005.

[11] Blender, "http://www.blender.org." A Suite of Tools for 3D Creation: Last accessed August 2009.

[12] Bluethmann, W., Ambrose, R. O., Diftler, M. A., Askew, R. S., Huber, E., Goza, M., Rehnmark, F., Lovchik, C., and Magruder, D., "Robonaut: A robot designed to work with humans in space.," *Auton. Robots*, vol. 14, no. 2-3, pp. 179–197, 2003.

[13] Breazeal, C., *Learning by Scaffolding*. PhD thesis, M.I.T. Department of Electrical Engineering and Computer Science, 1998.

[14] Breazeal, C., "Towards sociable robots," *Robotics and Autonomous Systems*, vol. 42, pp. 167–175, Jan. 2003.

[15] Bruemmer, D. J., Boring, R. L., Few, D. A., Marble, J. L., and Walton, M. C., ""I call shotgun!": an evaluation of mixed-initiative control for novice users of a search and rescue robot.," in *Proc. IEEE Conference on Systems, Man, and Cybernetics*, pp. 2847–2852, 2004.

[16] Capek, K., *Rossumovi univerzln roboti*. unknown, 1922.

[17] Charles Stark Draper Laboratory, Cambridge, Massachusetts, *The Timeliner User Interface Language (UIL) System for the International Space Station*.

[18] Chernova, S. and Veloso, M., "Confidence-based policy learning from demonstration using gaussian mixture models," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*, May 2007.

[19] Collett, T. H. J., Macdonald, B. A., and Gerkey, B. P., "Player 2.0: Toward a practical robot programming framework," in *Proc. of the Australasian Conf. on Robotics and Automation (ACRA)*, (Sydney, Australia), 2005.

[20] Dilman, I., *Free Will: An Historical and Philosophical Introduction*. Routledge, 1999.

[21] Dorsey, J., *Continuous and Discrete Control Systems: Modeling, Identification, Design and Implementation*. New York: McGraw-Hill, 2002.

[22] Drury, J. L., Scholtz, J., and Yanco, H. A., "Awareness in human-robot interactions," in *In Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, pp. 111–119, 2003.

[23] Egerstedt, M., Johansson, K., Lygeros, J., and Sastry, S., "Behavior based robotics using regularized hybrid automata," in *Proc. IEEE Conference on Decision and Control*, (Phoenix, Arizona), Dec. 1999.

[24] Fasano, G. and Franceschini, A., "A multidimensional version of the kolmogorov-smirnov test," *Monthly Notices of the Royal Astronomical Society*, vol. 225, pp. 155–170, 1987.

[25] Fong, T. W., Nourbakhsh, I., and Dautenhahn, K., "A survey of socially interactive robots," *Robotics and Autonomous Systems*, 2003.

[26] Fong, T. W., Thorpe, C., and Baur, C., "Collaboration, dialogue, and human-robot interaction," in *Proceedings of the 10th International Symposium of Robotics Research*, (London), Springer-Verlag, November 2001.

[27] Fong, T. W., Thorpe, C., and Baur, C., "Advanced interfaces for vehicle teleoperation: Collaborative control, sensor fusion displays, and remote driving tools," *Autonomous Robots*, vol. 11, pp. 77–85, July 2001.

[28] Fritzke, B., "Growing grid - a self-organizing network with constant neighborhood range and adaptation strength," *Neural Processing Letters*, vol. 2, pp. 9–13, 1995.

[29] Geppert, L., "Qrio, the robot that could," *IEEE Spectrum*, vol. 41, pp. 34–37, May 2004.

[30] Gerkey, B. P., Vaughan, R. T., and Howard, A., "The player/stage project: Tools for multi-robot and distributed sensor systems," in *ICAR 2003*, (Coimbra, Portugal), pp. 317–323, June 2003.

[31] Gerkey, B. P., Vaughan, R. T., and Howard, A., "The player/stage project: Tools for multi-robot and distributed sensor systems," in *In Proceedings of the 11th International Conference on Advanced Robotics*, pp. 317–323, 2003.

[32] Gockley, R., Bruce, A., Forlizzi, J., Michalowski, M. P., Mundell, A., Rosenthal, S., Sellner, B. P., Simmons, R., Snipes, K., Schultz, A., and Wang, J., "Designing robots for long-term social interaction," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pp. 2199 – 2204, IEEE, August 2005.

[33] Goldberg, K., Dreyfus, H., Goldman, A., Grau, O., Gržinić, M., Hannaford, B., Idinopulos, M., Jay, M., Kac, E., and Kusahara, M., eds., *The robot in the garden: telerobotics and telepistemology in the age of the Internet*. Cambridge, MA, USA: MIT Press, 2000.

[34] Gray, R. M., *Entropy and Information Theory*. New York: Springer-Verlag, 1991.

[35] Grollman, D. H. and Jenkins, O. C., "Dogged learning for robots," in *IEEE International Conference on Robotics and Automation*, pp. 2483–2488, April 2007.

[36] Grollman, D. H., Jenkins, O. C., and Wood, F., "Discovering natural kinds of robot sensory experiences in unstructured environments," in *NIPS 2005 Workshop on Machine Learning Based Robotics in Unstructured Environments*, (Whistler, British Columbia), December 2005.

[37] Harlan, R. M. and McClarigan, S., "Creating emergent behaviors: two robotics labs that combine reactive behaviors," in *SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*, (New York, NY, USA), pp. 441–445, ACM Press, 2005.

[38] Howard, A. M. and Park, C. H., "Haptically guided teleoperation for learning manipulation tasks," in *Proceedings of Robotics: Science and Systems: Workshop on Robot Manipulation*, (Atlanta, GA), 2007.

[39] Howard, A. M. and Paul, W., "A 3d virtual environment for exploratory learning in mobile robot control," in *Proc. IEEE Conference on Systems, Man, and Cybernetics*, (Hawaii), pp. 306–310, Oct. 2005.

[40] IFR Statistical Department, "Executive summary of world robotics 2005," 2005.

[41] Kac, E., "Origin and development of robotic art," *Art Journal. Digital Reflections: The Dialogue of Art and Technology, Special issue on Electronic Art*, vol. 56, no. 3, pp. 60–67, 1997.

[42] Kadous, M. W., Sheh, R. K., and Sammut, C., "Autonomous traversal of rough terrain using behavioural cloning," in *Proceedings of the International Conference on Autonomous Robots and Automation*, (Palmerston North, New Zealand), 2006.

[43] Keogh, E. J. and Pazzani, M. J., "Derivative dynamic time warping," in *In First SIAM International Conference on Data Mining (SDM2001*, 2001.

[44] Koenig, S. and Simmons, R., "Unsupervised learning of probabilistic models for robot navigation," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation (ICRA '96)*, pp. 2301 – 2308, 1996.

[45] Kohonen, T., *Self-Organizing Maps*, vol. 30 of *Springer Series in Information Sciences*. Springer Verlag, third ed., 2001. ISBN 3–540–67921–9, ISSN 0720–678X.

[46] Krohling, R. A., Jaschek, H., and Rey, J. P., "Designing pi/pid controllers for a motion control system based on genetic algorithms," in *Proc. IEEE International Symposium on Intelligent Control* (Ciliz, K. and Istefanopulos, Y., eds.), (New York, NY, USA), pp. 125–130, July 1997.

[47] Law, A. M. and Kelton, D. W., *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 2000.

[48] Lima, P. U., Custdio, L. M. M., Akin, H. L., Jacoff, A., Kraetzschmar, G. K., Kiat, N. B., Obst, O., Rfer, T., Takahashi, Y., and Zhou, C., "Robocup 2004 competitions and symposium: A small kick for robots, a giant score for science.," *AI Magazine*, vol. 26, no. 2, pp. 36–61, 2005.

[49] Liu, J., Dukes, I., and Hu, H., "Novel mechatronics design for a robotic fish," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Edmonton, Canada), 2005.

[50] Ma, X., Sun, Z., and He, Y., "Analysis and design of fuzzy controller and fuzzy observer," *IEEE Transactions on Fuzzy Systems*, vol. 6, pp. 41–51, February 1998.

[51] Mataric, M. J., "Designing and understanding adaptive group behavior," *Adaptive Behaviour*, vol. 4, no. 1, pp. 51–80, 1995.

[52] Mataric, M. J., "Getting humanoids to move and imitate," *IEEE Intelligent Systems*, vol. 15, no. 4, pp. 18–24, 2000.

[53] Maye, A., Hsieh, C., Sugihara, G., and Brembs, B., "Order in spontaneous behavior," in *Public Library of Science*, May 2007.

[54] Mayer-Kress, G. J., Newell, K. M., and Liu, Y.-T., "What can we learn from learning curves?," in *International Conference on Complex Systems*, (Nashua, NH), 1998.

[55] Metchev, S. A. and Grindlay, J. E., "A two-dimensional kolmogorov-smirnov test for crowded field source detection: Rosat sources in ngc 6397," *Monthly Notices of the Royal Astronomical Society*, vol. 335, p. 73, 2002.

[56] Michael Dittenbach, Andreas Rauber, D. M., "Uncovering hierarchical structure in data using the growing hierarchical self-organizing map," *Neurocomputing*, vol. 48, pp. 199–216, October 2002.

[57] Michels, J., Saxena, A., and Ng, A. Y., "High speed obstacle avoidance using monocular vision and reinforcement learning," in *ICML '05: Proceedings of the 22nd international conference on Machine learning*, (New York, NY, USA), pp. 593–600, ACM, 2005.

[58] Miller, L. H., "Table of percentage points of kolmogorov statistics," *Journal of the American Statistical Association*, vol. 51, no. 273, pp. 111–121, 1956.

[59] Miller, T. G., Bretl, T. W., and Rock, S., "Control of a climbing robot using real-time convex optimization," in *IFAC Symposium on Mechatronic Systems*, (Heidelberg, Germany), 2006.

[60] Minato, T., Shimada, M., Itakura, S., Lee, K., and Ishiguro, H., "Evaluating the human likeness of an android by comparing gaze behaviors elicited by the android and a person," *Advanced Robotics*, vol. 20, no. 10, pp. 1147–1163, 2006.

[61] Murphy, R. R., *An Introduction to AI Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, Nov 2000.

[62] Murphy, R. R. and Burke, J. L., "Up from the rubble: lessons learned about hri from search and rescue," in *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, pp. 2199 – 2204, August 2005.

[63] Nichols, B., Buttlar, D., and Farrell, J. P., *Pthreads Programming*. 101 Morris Street, Sebastopol, CA 95472: O'Reilly, 1998.

[64] Nilsson, T., "KiKS is a khepera simulator," Master's thesis, Ume University, Sweden, 2001.

[65] NIST/SEMATECH, "NIST/SEMATECH e-Handbook of Statistical Methods." http://www.itl.nist.gov/div898/handbook/: Last accessed November 2009.

[66] Oates, T., Schmill, M. D., and Cohen, P. R., "A method for clustering the experiences of a mobile robot that accords with human judgments," in *in Proceedings of IJCAI*, pp. 846–851, 2000.

[67] O'Connell, J., "Prepared remarks of state superintendent of public instruction," February 2006, (Sacramento, CA).

[68] oriented Graphics Rendering Engine, O., "www.ogre3d.org." An Open Source Rendering Engine. Last accessed August 2009.

[69] Peacock, J. A., "Two-dimensional goodness-of-fit testing in astronomy," *Monthly Notices of the Royal Astronomical Society*, vol. 202, pp. 615–627, 1983.

[70] Pew, R. W. and Mavor, A. S., eds., *Representing Human Behavior in Military Simulations: Interim Report*. The National Academies Press, 1997.

[71] Player-svn, "https://playerstage.svn.sourceforge.net/svnroot/playerstage/." The Subversion Repository for the Player Project. Last accessed February 2009.

[72] Pollock, J. L., *Thinking about Acting: Logical Foundations for Rational Decision Making*. New York: Oxford University Press, 2006.

[73] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical recipes in C (2nd ed.): the art of scientific computing*. New York, NY, USA: Cambridge University Press, 1992.

[74] Remy, S. and Howard, A. M., "Learning approaches applied to human-robot interaction for space missions," *Intelligent Automation and Soft Computing*, 2008. to appear.

[75] Remy, S. and Howard, A. M., "The use of coherence in learning behaviors via teleoperation," in *Proceedings of the International Symposium on Robot and Human Interactive Communication (RO-MAN 08)*, (Munich, Germany), 2008.

[76] Remy, S. and Howard, A. M., "Predicting the robot learning curve based on properties of human interaction," in *AAAI Spring Symposium Series 2009: Agents that Learn from Human Teachers*, 2009.

[77] Remy, S., Park, C. H., and Howard, A. M., "Improving the performance of ann training with an unsupervised filtering method," in *2009 International Joint International Conference on Neural Networks*, 2009.

[78] Ritter, F. E. and Schooler, L. J., "The learning curve.," in *International Encyclopedia of the Social and Behavioral Sciences*, Amsterdam: Pergamon, Elsevier Science, 2002.

[79] Roessingh, J. J. M. and Hilburn, B. G., "The power law of practice in adaptive training applications," 2000.

[80] Salcudean, S., Zhu, W., Abolmaesumi, P., Bachmann, S., and Lawrence, P., "A robot system for medical ultrasound," in *Proceedings 9th International Symposium of Robotics Research (ISRR'99)*, 2000.

[81] Sarle, W. S., "Neural networks and statistical models," in *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, (Cary, NC), pp. 1538–1550, SAS Institute, April 1994.

[82] Saunders, J., Nehaniv, C. L., and Dautenhahn, K., "Teaching robots by moulding behavior and scaffolding the environment," in *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, (New York, NY, USA), pp. 118–125, ACM Press, 2006.

[83] Saxena, A., Driemeyer, J., Kearns, J., Osondu, C., and Ng, A. Y., "Learning to grasp novel objects using vision," in *In 10th International Symposium of Experimental Robotics (ISER*, 2006.

[84] Saxena, A., Ng, A., and Chung, S., "Learning depth from single monocular images," *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005*, vol. 18, 2005.

[85] Scholtz, J., "Theory and evaluation of human-robot interaction," in *Hawaii International Conference on System Science*, vol. 36, 2003.

[86] Scholtz, J., "Creating synergistic cyberforces," *Multi-Robot Systems: From Swarms to Intelligent Automata*, 2002.

[87] Scholtz, J. and Bahrami, S., "Development of an evaluation methodology for the bystander role of interaction," in *Proc. IEEE Conference on System, Man, and Cybernetics*, 2004.

[88] Science and Technology Indicator Project Team, *Science and Technology Indicators: 2000*. Japan: National Institute of Science and Technology Policy (NISTEP) Science and Technology Agency, April 2001.

[89] Sehad, S. and Touzet, C., "Self-organizing map for reinforcement learning: obstacle-avoidance with khepera," in *From Perception to Action: the Right Direction? Proceedings "From Perception to Action" Conference*, (Los Alamitos,CA), pp. 420–423, IEEE Computer Society Press, 1994.

[90] Spergel, D., Piran, T., Loeb, A., Goodman, J., and Bahcall, J., "A simple model for neutrino cooling of the large magellanic cloud supernova.," *Science*, vol. 237, no. 4821, pp. 1471–1473, 1987.

[91] Steinfeld, A., Jenkins, O. C., and Scassellati, B., "The oz of wizard: simulating the human for interaction research," in *HRI '09: Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, (New York, NY, USA), pp. 101–108, ACM, 2009.

[92] Sutton, R. S. and Whitehead, S. D., "Online learning with random representations," in *International Conference on Machine Learning (ICML)* (Kaufmann, M., ed.), pp. 314–321, 1993.

[93] Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M. F., and Rother, C., "A comparative study of energy minimization methods for markov random fields," in *9th European Conference on Computer Vision*, vol. 3952 of *Lecture Notes in Computer Science*, pp. 16–29, Springer, May 2006.

[94] The Defense Advanced Research Projects Agency, "The DARPA Urban Challenge." Last accessed: August 2007.

[95] The Open Dynamics Engine, "http://www.ode.org." Open Source Library for Simulating Rigid Body Dynamics.

[96] Thrun, S. and others, "Stanley: The robot that won the darpa grand challenge," *Journal of Robotic Systems*, vol. 23, no. 9, pp. 661–692, 2006.

[97] Toh, E. K. and Wong, C. Y., "An expert system for real-time control of the sir-3 robotic system," in *IEEE International Sympoisum on Circuits and Systems*, 1991.

[98] Trinkle, J., "A quasistatic analysis of dexterous manipulation with sliding and rolling contacts," in *IEEE International Conference on Robotics and Automation*, May 1989.

[99] Unknown, "Self organizing map (SOM) C++ class and demo 1.0."

[100] Urting, D. and Berbers, Y., "Marineblue: A low-cost chess robot," in *Robotics and Applications*, pp. 76–81, IASTED International Conference Robotics and Applications, RA 2003, June 25-27, 2003, Salzburg, Austria, 2003.

[101] Vijayakumar, S., D'Souza, A., Shibata, T., Conradt, J., and Schaal, S., "statistical learning for humanoid robots," *autonomous robots*, no. 1, pp. 59–72, 2002.

[102] Webots, "http://www.cyberbotics.com." Commercial Mobile Robot Simulation Software. Last accessed February 2008.

[103] Winston, W. L., *Introduction to Mathematical Programming*. Belmont, CA: Duxbury Press, 1995.

[104] Wong, H. C., Bern, M., and Goldberg, D., "An image signature for any kind of image."

[105] Wood, D., Bruner, J. S., and Ross, G., "The role of tutoring in problem-solving," *Journal of Child Psychology and Psychiatry*, vol. 17, pp. 89–100, 1975.

[106] Wray, R. and Laird, J., "Variability in human behavior modeling for military simulations," in *Proc. of the Conference on Behavior Representation in Modeling and Simulation*, (Scottsdale, AZ), 2003.